

DIVISION Power Systems
TM 7975:70-639
DATE 15 September 1970
W.O. 1475-78-2000

TECHNICAL MEMORANDUM

AUTHOR(S): N. K. Rumpf

TITLE: Program Description of the Modified Thermal Analyzer
Digital Computer Program

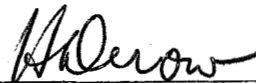
ABSTRACT

The Thermal Analyzer Program (TAP) has undergone a series of modifications for operation on the IBM 360/65. These modifications have increased its flexibility to handle a variety of heat transfer problems, reduced the cost of running the program, reconstructed the program layout to facilitate further changes, and streamlined the data input to the program. The programming performed to achieve these results is described. Subroutine level flow charts are included along with a program listing. This document acts as a supplement to TM 4942:69:602 and describes the program from a programmers point of view.

KEY WORDS: program description, computer program, thermal analysis,
transient, steady state, heat transfer

APPROVED:

DEPARTMENT HEAD



H. Derow



AEROJET-GENERAL CORPORATION

PROGRAM DESCRIPTION OF THE MODIFIED THERMAL ANALYZER
DIGITAL COMPUTER PROGRAM

1.0 INTRODUCTION

The Thermal Analyzer Program (TAP) was originally conceived and written at Lockheed, Sunnyvale. Variations of the original program have been made and have come into fairly wide use in the aerospace industry. The current modifications were made with several thoughts in mind: (1) expand the flexibility of the program so that it is more capable of analyzing complete systems; (2) provide a nuclear heating capability compatible with SNAP-8 needs; (3) reconstruct the program in a modular form to facilitate further modifications should they be desired; (4) minimize the amount of core storage used since the program is being run on a time shared machine and computer charges depend on the amount of core used; (5) examine and modify the iteration scheme to increase running speed of the program; (6) modify the input to make the program easier to use and to provide the user with additional flexibility so that he, by using his own ingenuity, could achieve the same results with reduced computer costs or could expand the amount of information he could extract from any computer run; (7) modify the input routines to perform more checking of data to prevent costly, erroneous runs. In each case the intended alterations met with at least some measure of success. In some cases the achieved results exceeded original expectations by a significant amount. It should be pointed out that much of the success of the current work was possible because of a good original concept and organization of the basic solution technique.

The current version of the program has been described by D. E. Knittle (Reference 1). It is the intent of this report to supplement that description by providing information of interest to the programmer. To do this, the specific changes made to the original program are described in some detail. The main computer program and each of its subroutines is described in sufficient detail to permit the reader to follow the path of the program and understand both the reason for the existence of section of programming and its relationship to other sections of the program. A detailed description of the input is also given here, despite the fact that this is well covered in Reference 1. When that document was being written several final changes were in progress. During the

programming effort, it was found that some of the intended alterations were inadvisable, at least at this time. Since the input is not precisely the same as previously described, it was felt advisable to rewrite this entire section rather than just giving the exceptions to the existing documentation. This is done in the description of subroutine FIN.

2.0 THE PROGRAM AND ITS MODIFICATIONS

2.1 AN OVERVIEW OF TAP CONSTRUCTION

The structure of TAP is of the form of a main program which performs the primary calculations and also acts as a calling program to permit the activation of a group of subroutines to perform the secondary calculations. Since the program is intended to solve a diverse range of heat transfer problems it is necessary to provide the capability to perform a variety of calculations without requiring that each of them be performed on each pass through the program. This can be achieved by providing sufficient flexibility in the input data to permit the user to control the calculations to be performed. In the TAP program the data consists of required data, in the form of thermal conductance, capacitance and initial temperatures, and of optional data, in the form of functions and tables which are called by the main program only when required to solve the problem at hand. The first task in the program is to read in the required data and then read in the optional data and set appropriate flags for each option chosen.

TAP uses a finite element approach to solve transient heat transfer problems. A full explanation of its capabilities and the equations used to perform the calculations are described in Sections 3 and 4 of Reference 1. A full description of each routine used in the program and its function are given later in this report. The program first calculates the time step to be used in the next iteration. This is calculated to find the maximum time step possible without causing an instability in the numerical solution of the transient partial differential heat transfer equations. If a maximum time step is chosen by the user then this becomes the time step instead of the one calculated, provided a conflict with the stability criterion does not exist.

If fluid flow calculations have been specified in the data these calculations are performed. A check is then made to determine whether results are

to be printed after the current time step; if it is, a flag is set. Next, the nuclear heating calculations are called for if specified in the data. The calculation of nodal temperatures (primary calculations in the program) occur next. Options are then tested for a specialized fluid flow routine, for a steady-state check, and for a fluid flow temperature averaging scheme. If the print flag is set a print is obtained; if not, options are tested for use of the tables, radiation heat transfer calculations, aerodynamic heating calculations, cathode follower calculations, and latent heat calculations. The end of the transient is then tested for and if not found the calculations continue.

All of the calculations which can be called for data options are contained within separate subroutines where there are a sufficient number of calculations to justify a subroutine. Most of these are specified in the FIN program by an explicit function number. Function data is passed between subroutines through commons (FUN1, FUN2, etc.). Other information is passed as arguments to the subroutines.

Only one fundamental change in the program has been made and should be noted: wherever possible, thermal resistances which appeared in the original have been replaced by conductances for the purpose of internal calculations. Resistances are denoted by R and conductances by COND in the program nomenclature. Parallel resistance paths are added as a sum of reciprocals. These same paths require only addition when handled as conductances. This change significantly reduces the number of divisions required to perform the arithmetic and consequently speeds the run time of the program.

2.2 PROGRAM FLEXIBILITY AND NUCLEAR HEATING

Before alterations were made, the TAP program was most applicable for simple conductive networks. Through the intelligent use of this network it was possible to also perform some limited fluid flow calculations. Within these limitations the program was quite flexible permitting various types of heat input, radiative and limited convective heat transfer, and the ability to vary resistors, capacitors, temperatures, or heat inputs as a function of time, or as functions of each other by using the tables. The program also contained a subroutine to calculate heat input to a node in an arbitrary manner by the user.

The arbitrary heat input subroutine was removed since it was determined to be of limited utility. In its place, a nuclear heating program was constructed which was consistent with the design of the SNAP-8 reactor. The equations used in this subroutine are given as 2.7.8 to 2.7.11. Equation 2.7.8 is the neutron flux equation for a single group of delayed neutrons and is derived making the assumptions that the reactor has been at steady state for a sufficient period of time for all the precursor groups to be at equilibrium and that the reactivity is about 2.5×10^{-3} or less, i.e. $(\beta - \rho + \lambda\lambda)^2 \gg |2\lambda\lambda\rho|$.

Equation 2.7.9 is a simplified equation for decay heat. Under normal operating conditions the beta and gamma heating is approximately 6% of the total heat generated. This rapidly drops to about 3% if the reactor is shut down. These facts determine the following method of calculating reactor heating. If the power is at or near the base power, the beta and gamma heating closely follows the neutron heating and Equation 2.7.8 is used. If the reactor is shut down Equation 2.7.9 is used. Between these two extremes a combination of the two equations is used. Let us say that the power is dropped to 50% of the base; the beta-gamma heating should drop off rapidly and the assumption made that all the heating is neutron heating is still a valid one. Now consider the case where the neutron power drops to less than 50% of base power. It is unlikely that this would happen slowly since the reactor is designed to operate at a constant base power level. If it should, then the beta-gamma heating would decay with the power level. Should the power drop rapidly, then the beta-gamma heating becomes significant when the neutron power approaches the beta-gamma heating in level. When neutron power is less than .06 of the base power then the beta-gamma heating is added to 0.94 of the neutron power (that portion actually supplied by neutrons and prompt gammas at full power; the remaining .06 is always provided by the decay betas and gammas). This determines the use of Equation 2.7.10.

Additional flexibility has been given to the program with the implementation of fluid flow capability. Three functions control the use of this capability; Functions 10, 11, and 12. Function 10 provides the basic capability. Heat is transferred by fluid flow from node to node. This capability can be invoked simply by specifying the nodes between which the fluid is flowing, by indexing this information to the table specifying the w_{c_p} as a function of time,

and by specifying the thermal capacitance of the node. Function 11 is used to combine the flow from two or more nodes into a single node and average the temperature. This would be used at the junction point of two or more pipes. Function 12 serves the same purpose as Function 10 except that the fluid is moved through a series of nodes in each iteration. The intent of Function 12 is to replace Function 10 when the use of Function 10 would result in the time steps being significantly reduced in size (which causes a proportionate increase in the computer run time).

One other capability exists in the fluid flow functions. It is possible to leave a fluid node empty until some specified time during the calculations, after which the transfer of heat by fluid flow through the node indicated takes place. This option is also specified in Function 10.

2.3 MODULARIZING THE PROGRAM

The intent of organizing the program in modular form is to separate various sections of programming along functional lines. This facilitates future modification since sections of programming can be removed and replaced intact. It also makes available one common functional base which may be addressed from different sections of the program. This minimizes the need for similar sections of coding to appear at several different locations in the program.

The program as conceived was constructed in good modular forms. It remained to follow the same practices in placing the new coding into the program. The original arbitrary heating subroutine was replaced with the nuclear heating subroutine. Both Function 5, which describes the nuclear heating data, and Function 13, which describes the feedback coefficients and the dead-band temperature data, are used in the nuclear heating subroutine, HEATIN. Functions 10 and 12 are used in subroutine FLOW. Although the calculations called out by these two functions are somewhat different, their basic purpose is the same and they were, therefore, placed together. Subroutine TAVG contains the averaging scheme to mix two or more nodes into a common node and average the temperature. Function 11 calls out this subroutine. Finally a subroutine to test for equilibrium of the entire system of nodes during a transient is separated as subroutine STEADY. In each case, these subroutines could be modified or replaced with a minimal effect on other portions of the program.

A more detailed description of each subroutine, including its ties to other parts of the program, is given later in this report.

One change which was made from the original program was to place the function information into a series of common blocks called FUN1, FUN2, etc. This significantly reduced the number of arguments being passed between certain subroutines and, at the same time, placed all the function information into readily definable blocks. Since almost all the function information is used in modular form, the placement of the function information into discrete modules was also considered advisable.

2.4 CHANGES IN CORE STORAGE

The modifications to the TAP program were undertaken with the intent that the program would be run on an IBM 360 machine. The 360 series provide for a multi-programming environment and computer charges are based on the percentage of core used as well as the actual run time. Because it was possible to have temperatures of 1000° and more, and because of the possibility of small time steps with slowly changing temperatures, it is possible to lose significant accuracy during a transient by using single precision (about 6-1/2 decimal digits of accuracy). Consequently, all quantities used directly in the temperature calculations were typed REAL*8. All other floating point variables were left in single precision. This change increased the core storage. Another increase occurred when it was found advisable to increase the number of conductors (resistors) in the program from 700 to 999. The number of conductors is always larger than the number of nodes (unless the network is a closed circle). In two dimensions the ratio of conductors to nodes is usually in the range 1.5 to 2.0. In three dimensions this ratio is usually 2.0 to 4.0.

Minimization of core storage is partially achieved by not typing all real variables REAL*8. It is further achieved by condensing the integer variables by typing them INTEGER*2. This permits the use of integer numbers with values up to 65,535, which is well beyond any values which would be used in the program. The variables chosen to be typed in this manner are those that are dimensioned. It is here that the greatest savings are to be made. The other integer variables could also be typed INTEGER*2, but in this case ^{care} care would also have to be taken that they appeared at the end of COMMON lists to avoid

the problem of word boundary misalignment in core. Still another method of core reduction was to critically evaluate all dimensions to eliminate over-specification and to evaluate all coding to eliminate those sections which were no longer necessary for the program to achieve its objectives (one such section was tied to a variable dimensional 5000 which was no longer necessary).

A final method of core reduction was to overlay the program. Only one overlay can be intelligently done and that is to place the input routines, which are used only once, in a separate segment. This segment is composed of subroutines FIN and TABIN, which are input routines as well as READ and READ2, which are called only from the input routines.

The result of all these changes was to reduce the size of the program from just under 200K bytes to less than 150K bytes. It should be mentioned here that two other core reduction mechanisms are readily available. One simple one is to reduce the number of nodes from 700 to 600. It is highly unlikely that more than 600 nodes would be used even if all 999 conductors are used. The variables T, C, Q, RI, TR, IRCVAR and RIFIX could all be reduced in size from 700 to 600. This could also be done with some of the print specifications. Since most of these variables are typed REAL*8 the savings would be significant (more than 5K bytes). The change is not being made at this time because not enough physical cases have been run to show conclusively that this change is advisable. A second change would be slightly more difficult to make. The tables are currently dimensional $10^4 \times 30$ (12,480) bytes. All 30 tables are almost never used. Even if they should be used, all 10^4 positions in all the 30 tables will certainly not be used. It should be possible to halve the size of this array, singly dimension it, and set up a table of pointers to point to the location of the first element of each of the 30 tables.

2.5 MODIFICATION OF THE INPUT AND OUTPUT

Input to the TAP program is through three subroutines, TRCIN, which reads in the conductor, temperature and capacitor data; FIN, which reads in the function data, and TABIN, which reads in the tables. A small utility subroutine with two entry points (READ and READ2) is used by these input routines. These read data in either of two standard input formats.

x The first input change was to the first card read in by the program, the title card. Columns 1-5 of every run must read "TITLE." The actual run title begins in Column 7. An option was included to punch the temperature-capacitance cards at the completion of the transient by placing a "1" in Column 6 of the title card. It is standard practice to make a steady-state run before performing a transient analysis. This is because most models which must be solved by computer analysis are sufficiently complicated that the initial temperature distribution is not known. The steady-state run determines these temperatures and punches these out so that there is no need to keypunch this part of the data before running the transients.

Subroutine TRCIN originally read in first the resistor data, the temperature data and the capacitor data, in that order, with a -1 card at the end of each section used as a flag to proceed to the next section. The first change was to combine the temperature and capacitance cards to put the information on one card. Since both of these quantities are specified according to node number, then the card containing the node number can be used to specify both quantities.

The next change was to permit an easier method of specifying sequential conductor or node numbers if the value of the conductance or of the temperature and capacitance is identical. The method chosen to do this was to flag certain of the input fields by negative numbers. Thus, in specifying the input for temperature and capacitance, if a sequentially numbered group of nodes have the same temperature and capacitance, the placement of a -1 in Columns 9 and 10 of the first card of the sequence will result in the internal numbering of all nodes up to and including the last of the sequence. This means that only two cards, the first and last of the sequence, must be specified to define a sequential series of nodes if the first card is flagged with a -1.

The same type of sequential numbering scheme has been programmed for the conductor data. In this case not only must the conductors be sequentially numbered, but the nodes to which they are connected must also be sequential. Prior knowledge that this capability exists makes it easy to take advantage of it when assembling the nodal network. Additional time saving devices are available in the conductor block by proper placement of negative signs. The

sequential numbering of conductors and nodes in this block occurs if negative signs are placed in front of both nodes of the first card in the sequence. If only one of the nodes is made negative that series of nodes and the conductors are sequenced. The other node number is kept constant during the sequencing. This model is representative of several nodes connected to a large heat sink. A detailed description of the input is given in Section 3.2 of this report.

Subroutine TRCIN has also been modified to perform some checking of the input data. If the maximum conductor number exceeds 999 or if the maximum node number exceeds 700 the run is terminated with an appropriate error message. A check is made to assure that no conductor is attached to a non-existent node. If that condition exists a flag is set and the run terminated at the completion of the input checking. Next each node is checked to see that it is connected to a conductor; if not, another error message is printed and a flag set. If a negative capacitance exists for a non-connected node, then it is possible for the node to remain at a constant temperature throughout the problem. Even though this condition provides no useful information, the condition is permitted.

The next check of the input data is to flag the variable conductors. Those that are constant require no further calculations during the transient. Flagging the variable conductors saves computer time by minimizing the calculations required during the transient.

Input to subroutine FIN, to read in the function data, has also been modified to reduce the number of cards necessary to specify the function information desired as well as to perform some input checking.

Functions 2, 8, 10 and 12 have been programmed so that a negative number in Columns 11-15 will act as a flag for sequential numbering. In each case the flag card acts as the initial card in the sequence while the card immediately following it acts as the last in the sequence. A more thorough description of this capability is given in Section 3.3 of this report.

Functions 1-6, 8, and 10-12 are each checked for the amount of input given to their respective functions. If the input exceeds the amount dimensioned in the subroutine an error message is printed and the run aborted. An error of this type could do serious damage to the calculation procedure. In each function where a conductor or node number is referenced, a check is made to be

certain that it exists. If not, an error message is printed, a flag is set, and the run is aborted after the remainder of the data has been checked.

Function 9 input has been modified to provide additional flexibility for the user. Problem constants 1 and 2 in Function 9 previously permitted a single specification for the time interval (if the user chose to over-ride the program calculations) and for the print interval. It is now possible to specify up to 5 intervals during different periods of the transient.

In specifying the print interval it is sometimes desirable to examine output at frequent intervals during a particular stage of the transient. It is now possible to do this and then change to a longer interval to avoid the excessive printout that would result if the shorter interval were continued throughout the transient. The time interval is often specified when the user wishes to examine the changes in temperature resulting from some rapid change specified in one of the tables (usually heat input). He shortens the time interval and the print interval to examine the change in detail. Later in the transient he increases both, permitting the program to calculate its own time interval and shorten the run time of the program.

Other checks are also made in subroutine FIN. Function 8 must be specified or no printed output will result. If Function 8 is omitted, the run is aborted. Checks are also made to be certain that the print interval has been specified (constant 1 of Function 9) and that the problem completion time has been specified (constant 4 of Function 9).

Another change to the input has been the inclusion of a second standard card reading routine which has been included as entry READ2 in subroutine READ. This additional capability consists of the ability to read in two fields of floating point data instead of only one as was previously the case. This serves the purpose of reducing the number of data cards necessary to specify the problem.

The output format for the program has been modified to provide easier user access to the data. Before modification the method of data printout was to first print a format of the output with seven pieces of data per line of line, e.g.:

T(26)	T(27)	T(106)	T(154)	Q(27)	C(106)	T(154)
-------	-------	--------	--------	-------	--------	--------

Thereafter, at each print step only the numbers were printed. It was necessary to remember which number referred to which quantity in the print format, or to continually refer back to the format. If a large number of quantities was being printed this procedure became quite cumbersome.

The current print scheme is to print only five pieces of data per line, with the identification of each quantity printed whenever the number is printed. The FORTRAN output format has been retained as E15.5 for each number printed. In addition, if the nuclear heating capability is used, the δK of the reactor is also printed at each interval. A sample of the output can be found as Figure 10.7 of Reference 1.

2.6 MODIFYING TAP DIMENSIONS

The TAP program as described in this report will handle physical problems which can be modeled with up to 700 nodes and 999 conductors. This capability fits in approximately 150K bytes of an IBM 360 computer. A program of this size is sufficient to give an accurate model of most heat transfer systems that will be encountered. Should additional capacity be desired, the modifications to the program are fairly easy to accomplish. Also, if a smaller version of the program were desired so that it could be run on a small computer, it is possible to reduce the dimensions and also to strip some sections from the program which are seldom used.

To expand the program, most of the changes are in the dimensions. In the main program the dimensions of quantities T, COND, TR, RI, RC, I2V, I3V, C, Q, PRUNT and PRINT would have to be changed. Since PRINT is equivalenced, its boundaries would also require change. In COMMON VARY, IRVAR would be changed as would the dimensioned variables in COMMON FUN8. All variables dimensioned 700 would change to the new node maximum while those dimensioned 999 would change to the new conductor maximum. The DO loops ending at Statements 7 and 8 would require similar changes. Immediately following Statement No. 291 various quantities are referenced for arbitrary interpolations. Where 999 is used as a multiplier it would be replaced by the maximum conductor number; where 299 is used, it would be replaced by the difference of the maximum conductor and node numbers. These same changes would be made in the print section immediately following Statement No. 400. In the other subroutines changes in dimensions would have to be made as follows:

<u>SUBROUTINE</u>	<u>QUANTITIES</u>
TRCIN	IRVAR, I2V, I3V, COND, T, C, RI, TR
FIN	T, I2V, / FUN8/
TABIN	IRVAR
FLOW	T, C, Q, TR, RI
HEATIN	T, Q
TAVG	T, C, RI
STEADY	T, TSAVE

Also in subroutine TRCIN the DO loop ending at Statement 1 must be changed to fit the dimensions of I2V. Statements 50 and 80 must have their dimensions changed to reflect the nodal and conductor changes.

To obtain a smaller program, say one with 100 nodes and 150 conductors, all the changes noted to increase the program size would be made. In addition, it is recommended that subroutines HEATIN, TAVG, LAT, QECK, and QRW be removed as well as entry FSTFLO in subroutine FLOW. This would also permit elimination of Functions 3, 5, 6, 11, 12 and 13. Functions 1 and 4 could also be removed. The number of tables could be reduced to 15 and pointers used to conserve space. Function 2 would be dimensioned 50; Function 7, 50; Function 8, 100; Function 9, unchanged; Function 10, 50; and Function 14, unchanged. Also, each of the sections in the main program which refer to these functions could be removed, further reducing program size. An ultimate size of less than 50K bytes is anticipated. Section 3.1 of this report describes the main program in detail.

2.7 EQUATIONS REFERENCED IN THE REPORT

The following equations are those referenced within this document and are taken from Reference 1. Two identifications are given for each equation; the number as used in this document and the number in the referenced document (in parentheses).

$$T_{\theta+\Delta\theta,i} = \frac{\Delta\theta}{C_i} \left(\sum_{j=1}^n \frac{T_{\theta,j}}{R_{ij}} + Q_{\theta,i} - T_{\theta,i} \sum_{j=1}^n \frac{1}{R_{ij}} \right) + T_{\theta,i} \quad \begin{matrix} 2.7.1 \\ (4.1) \end{matrix}$$

$$C_i = C_{Si} + (wc_p)_{\theta,i} \Delta\theta \quad \frac{2.7.2}{(4.8)}$$

$$T_{\theta+\Delta\theta,i} = \sum_{j=1}^n \left(\frac{T_{\theta,j}/R_{ij} + Q_i}{1/R_{ij}} \right) \quad \frac{2.7.3}{(4.11)}$$

$$T_{\theta+\Delta\theta,i} = \frac{1}{(wc_p)_{\theta,i}} \left(\sum_{j=1}^n \frac{T_{\theta,j}}{R_{i,j}} + Q_{\theta,i} - T_{\theta,i-1} \sum_{j=1}^n \frac{1}{R_{ij}} \right) \quad \frac{2.7.4}{(6.2)}$$

$$+ T_{\theta,i-1}$$

$$T_{\theta+\Delta\theta,i} = \frac{\Delta\theta}{C_i} \left(\sum_{j=1}^n \frac{T_{\theta,j}}{R_{i,j}} + Q_{\theta,i} - \left[\frac{T_{\theta,i} (C_i - [wc_p]_{\theta,i} \Delta\theta) + [wc_p]_{\theta,i} \Delta\theta}{C_i} \right] \cdot \sum_{j=1}^n \frac{1}{R_{ij}} \right. \\ \left. + \left[\frac{T_{\theta,i} (C_i - [wc_p]_{\theta,i} \Delta\theta) + T_{\theta,i-1} [wc_p]_{\theta,i} \Delta\theta}{C_i} \right] \right) \quad \frac{2.7.5}{(6.3)}$$

$$T_{\theta+\Delta\theta,i} = \frac{1}{(wc_p)_{\theta,i}} \left(\sum_{j=1}^n \frac{T_{\theta,j}}{R_{ij}} + Q_{\theta,i} - \bar{T} \sum_{j=1}^n \frac{1}{R_{i,j}} + \bar{T} \right) \quad \frac{2.7.6}{(6.5)}$$

where

$$\bar{T} = \left[\frac{T_{\theta,l} (wc_p)_{\theta,l} + T_{\theta,m} (wc_p)_{\theta,m} + \dots + T_{\theta,n} (wc_p)_{\theta,n}}{(wc_p)_{\theta,i}} \right]$$

and

$$(wc_p)_{\theta,i} = (wc_p)_{\theta,l} + (wc_p)_{\theta,m} + \dots + (wc_p)_{\theta,n}$$

$$T_{\theta+\Delta\theta,i} = T_{\theta+\Delta\theta,j} \quad \frac{2.7.7}{(6.8)}$$

$$Q_{n,\theta+\Delta\theta,i} = (Q_o) (P.F.) \left[\frac{\beta}{\beta\rho} \exp \left\{ \left(\frac{\lambda\rho_\theta}{\beta-\rho_\theta} \right) (\theta) \right\} - \right. \quad \frac{2.7.8}{(6.12)}$$

$$\left. \frac{\rho_\theta}{\beta-\rho_\theta} \exp \left\{ - \left(\frac{\beta-\rho_\theta}{\ell} \right) (\theta) \right\} \right]$$

$$Q_{\beta+\gamma,\theta+\Delta\theta,i} = 6.22 \times 10^{-2} (\theta)^{-0.2} (Q_o) (P.F.) \quad \frac{2.7.9}{(6.13)}$$

$$Q_{react,\theta+\Delta\theta,i} = 0.94 Q_{n,\theta+\Delta\theta,i} + Q_{\beta+\gamma,\theta+\Delta\theta,i} \quad \frac{2.7.10}{(6.14)}$$

$$Q_{react,\theta+\Delta\theta,i} = Q_{n,\theta+\Delta\theta,i} \quad \frac{2.7.11}{(6.15)}$$

$$h_i = \beta \left[\left(\frac{N_{Re}}{XM_\infty} \right)^{M_\infty} \right]^a (M_\infty)^b \frac{1}{T_R - T_\infty} \quad \frac{2.7.12}{(6.17)}$$

$$h_i = \beta \left[\frac{S \left(\frac{N_{Re}}{XM_\infty} \right)^{M_\infty}}{(T_L^1 / T_L)^{1.69}} \right] MN_{Pr}^{1/3} K \quad \frac{2.7.13}{(6.18)}$$

3.0 A DESCRIPTION OF EACH SUBPROGRAM

3.1 THE MAIN PROGRAM

The TAP program is constructed in two basic sections: (1) an initialization section where data is read, variables are initialized, flags are set and an initial data print is called for; (2) an iteration scheme where the various functions and tables, used in the problem specifications, are called for according to flags set in the initialization section and where the primary temperature calculations are performed.

The program begins by reading in a title card. Since multiple cases can be run, the program branches to Statement No. 1 at the end of each case. If a card with "TITLE" in Columns 1-5 is not found, additional cards are scanned until such a card is found or until all cards are exhausted. In addition, a flag is examined in Column 6 of the title card. If a "1" is found there, then at the end of the run the temperatures and capacitances are punched on cards suitable as input on a succeeding run.

Since the IBM-360 is a non-zeroed machine, it is necessary to assure that certain critical quantities are set to zero before the calculations begin. Some other quantities must be initialized to numbers other than zero. All this work is accomplished in the first section of programming.

The next section of the program covers the reading of data and the arrangement of the flow path for the program, based on the data read. The arranging is of two different types, both of which are designed to minimize the problem run time.

Based on the function input from subroutine FIN, assignments of statement numbers within the main program are made. These assignments are dependent on which of the functions is used. For any of the functions not used, a section of the main program may be skipped or one of the subroutine calls need not be made.

Based on the input from subroutine TRCIN, the second type of saving can be made. In this case, calculations on certain conductors need not be performed during each iteration. Flags for each conductor are set in entry RFIX of subroutine TRCIN, depending on whether the conductor remains fixed in value during the transient or whether it is variable.

After this work is completed, the temperatures of the zero capacitance nodes is calculated and the information called for in Function 8 is printed out. This information is for time zero in the transient. This action completes the initialization section and serves as the starting point for the iteration scheme.

The first step in the iteration is to save the time step from the iteration just completed so it may be available as information if a print is requested after the iteration is just starting. This occurs at Statement 290. Following this step there occur a series of calls to subprograms which establish the conditions of the nodal network in preparation for the primary temperature calculations.

X If tables have been used to describe the relation of one nodal quantity to another, an interpolation is used within the table to establish the current relationship. These calculations are performed only if the user specifies Function 7 in his input. If Function 2 is specified, then the next calculation is the conductance of those conductors where radiation affects the heat transfer. Similarly, use of Functions 4, 6, 1 and 3 would respectively call for calculations of convection conductors, aerodynamic heating, cathode follower, and latent heat information on the specified nodes. The choice of either the Eckert or the Ramo-Wooldridge method of aerodynamic heating is made in the Function 6 input. Use of the cathode follower is to permit the temperature of one node to be set equal to that of another node after each iteration.

After these calculations, a test for the end of the job is made before continuing with the iteration. The flag TOME is zero only during the first iteration and is used to note that iteration. If the problem completion time has not been exceeded, the iteration continues at Statement 205. If the run is complete, then the option to punch the temperature-capacitor cards is checked before returning to Statement 1 to check for the input of the next cast if one exists.

/ A check is made to be certain that the time step is not smaller than some minimum the user may specify in his input. Then if the current problem time does not exceed the problem completion time, the next iteration step is begun.

X The first step in the iteration is to determine the time step (DELTAT). The time constant (RC) of each node is calculated and the minimum noted. This serves as the first estimate of DELTAT. Next a check is made to determine if the user has specified a time step. If he has, it must be smaller than the minimum time constant (to avoid instability), and the lesser of the two numbers is used as DELTAT. The RC minimum may also be multiplied by some fraction to force an even smaller DELTAT on the iteration, thus, guaranteeing more accurate calculations for the relaxation procedure used as the solution technique. If the user does not specify a fraction, 1.0 is used. Finally, the minimum time step acceptable to the user is checked. If DELTAT is less than this, an error message is printed followed by a print of the current output data.

X The next calculations called for by the main program are the fluid flow calculations. These are called only if Function 10 was specified in the input. A determination is then made to establish whether a print of the output should occur at the completion of the current iteration. If it should, then the quantity PRTST is set to zero as the flag which is later tested. The next calculations are those for nuclear heating. These are performed only if Functions 5 and 13 were specified in the input.

At this point, the temperature calculations are begun. Equation 2.7.1 is the primary solution equation for the nodal temperatures. Calculations are performed only for nodes with a positive capacitance. The zero capacitance nodes are calculated next after a call to TRA to update the T/R and 1/R sums. Equation 2.7.3 is the one used in the calculations.

The final calculation performed is the one to pass fluid through a series of nodes in one time step. The use of Function 12 causes this calculation to be called. The last calculation step for the current iteration step is to update the problem time.

If the user chooses to check for steady state or equilibrium by specifying Function 14, then subroutine STEADY is called. If the steady-state criterion is met, then the output data is printed immediately. Subroutine TAVG is called if two or more fluid flow nodes are to be averaged into a single node. Function 11 is used to call this subroutine. Following this the PRTST flag is checked to see if a print is required on this iteration. If not the next iteration is begun immediately.

3.2 SUBROUTINE TRCIN

Subroutine TRCIN performs most of the secondary manipulations on temperatures, thermal capacitances and conductances. It is written in three sections. The first section reads in the conductance, temperature and capacitance data. The second section separates the fixed from the variable conductances and calculates the fixed data for permanent storage and later access. The third section calculates the $1/R$ and T/R sums used in the calculations for temperature change on each iteration step in the main program.

COMMON VARY in this subroutine stores information on the capacitor nodes and conductors which are variable. Most of the integer information is carried as INTEGER*2 to conserve space within the computer core. Several quantities are carried as REAL*8. These are used to permit the calculation of the temperature change with each time step in double precision arithmetic. Those calculations are performed in the main program. The subroutine also makes use of the READ and READ2 entries to subroutine READ. Each of these entries read in three fields of integer data. In READ this is followed by one E15.5 field and in READ2 by two E15.5 fields.

The initial section of the program reads in conductor or resistor data. The I1 field is used to specify the conductor number and the I2 and I3 fields are used to specify the two nodes to which each conductor is connected. The first E field contains the value of the conductance or resistance. The subroutine starts by placing a zero in the locations of all the node numbers appearing in the I2 field to which the conductors are connected. It next checks to see if the input is to be in the form of conductances or resistances. In the past, this program accepted only resistances. It now accepts conductances as well and this choice can be stipulated in two ways. If the first card in the conductor block reads COND in Columns 1-4 then the input is assumed to be in the form of conductances. Otherwise, input is assumed to be in the form of resistances, unless the resistor card is flagged with a C in Column 1. In this case that card only is assumed to be in the form of a conductance. The maximum conductor number is also determined in this section of programming so that future DO loops can stop when the index reaches the maximum conductor number. Except for the first card, all other cards are read (and printed) using the READ subroutine.

An option is available in reading the conductor cards which can significantly reduce the card input. If the node numbers in both the I2 and I3 fields are negative, the program will read the next card and internally generate all the intermediate sequential conductor and node numbers assigning the same conductance or resistance value, which appears on the last card, to each generated card. If only the I2 or the I3 field is negative, then the same thing occurs except the positive I2 or I3 field has its number remain constant during the internal generation. The last card in the conductor block contains only a -1 in the I1 field.

The conductor data are followed immediately by cards containing information on the thermal capacitances and the initial temperatures at each of the nodes. Input is read using the READ2 subroutine and is in the form of three I fields and two E15.5 fields. The node number is written in the first I field, the initial temperature is written in the first E field and the thermal capacitance is written in the second E field. If a -1 is inserted in the I2 field, it is possible to generate nodes internally in much the same manner as was done in the conductor input. After the card containing the -1, in the I2 field, the last card of a sequence of nodes is input. The subroutine internally generates the intermediate nodes, giving the same temperature and capacitance as appeared on the last card, to each. Any initial temperature which is input as zero in the temperature-capacitor block is given a value of 10^{-4} to facilitate a check of conductors and nodes to see that none of them remain unconnected. The last card in the temperature-capacitor block contains only a -1 in the I1 field. The maximum node number is also sought in this section of the programming so that future DO loops can stop when the index reaches that maximum number. After completion of the temperature-capacitor reading section, control is returned to the main program.

Subsequent to this, the check on unconnected conductor and nodes is performed. Two small DO loops perform this function. The first loop finds the two nodes to which each resistor is connected and notes whether the temperature of that node is other than zero. If not an error message is printed. At the same time it temporarily assigns a value of 1.0 to the $1/R$ sum (RI) of each node. The second DO loop then checks each node with a non-zero temperature to ascertain if RI for that node is also non-zero. If RI is

zero, and if the capacitance is non-negative, an error message is printed noting a node which is not connected to any conductor. The exception of the negative capacitance node permits the use of a node as a heat sink or as a leading node in one of the fluid flow routines.

The next section of the subroutine is entry point RFIX which is called from the main program prior to performing any temperature calculations (immediately before Statement No. 108). The purpose of this section is to distinguish between the variable and fixed resistors and also those nodes which are attached to variable resistors. The $1/R$ sum must be calculated for each node, to be used later in calculating temperature changes during each iteration. That portion of this sum which is from fixed resistors need be calculated only once during the program. This set of calculations is performed and the values stored as RFIX for each of the nodes to which a fixed conductance (resistance) is attached. The initial value of the $1/R$ sum is set equal to the fixed portion of that sum. The variable portion is added in the TRA section of this subroutine. Control is returned to the main program when these operations have been completed.

The final section of the subroutine begins at entry point TRA. In this section the quantities RI and TR are calculated for use in the main program. These calculations are performed twice; once to calculate temperature changes of all nodes with a positive capacitance and the second time to permit calculation of the zero capacitance nodes after the temperature of the surrounding positive capacitance nodes have already changed. The first entry is from Statement No. 205 in the main program while the second is from 255 of the main program. The only calculations which must be performed on the second pass are for TR. The argument IPASS directs the subroutine to perform only the necessary calculations.

If none of the conductors are variable, then it is not necessary to recalculate the RI's. Under these circumstances, the calculations at Statement Nos 105 and 122 are skipped. If IPASS is 2, then the RI also need not be calculated since neither the conductance nor RI are modified between the first and second entries to TRA. Consequently, the calculations at Statement No. 105 are skipped and control is directed to Statement No. 130 where RI is not

calculated and where only the zero capacitance nodes are considered when the TR calculations are repeated.

If IPASS is 1, then the RI's are reinitialized to their fixed portion. The TR's are initialized to zero and then the remaining RI and TR calculations are performed. The additional RI calculations need only be done on the variable resistors. The TR's are calculated for all nodes. Control is then returned to the main program.

3.3 SUBROUTINE FIN

Subroutine FIN is called only once from the main program immediately after the call to TRCIN. The sole purpose of this subroutine is to read in the function data which give the program its flexibility. Some of the functions recently placed in the program will be new to users familiar with older versions of TAP. Persons wishing to provide additional flexibility for the program can readily do so by creating new functions and factoring them into the calculation stream of the main program at the appropriate locations.

There are currently 14 functions in the program. Since the data read into these functions in FIN is generally used in one or two other locations, this information has been placed in a series of commons, each tied to its own function. Data for Function 1 is stored in COMMON FUN1, Function 2 is FUN2, etc. In addition, COMMON VARY contains data pertaining to those conductors which are variable. Flagging this information saves recalculation of those conductances which are fixed throughout the calculation of the transient. Much of the function data contains integer information, most of which is designated INTEGER*2 to conserve space in the computer core.

The FORMAT statements are listed separately at the front of the subroutine since some of them are referenced from more than one location. Also, at the front of the subroutine several quantities are initialized in anticipation of use in the input of various functions. Data are read in by making use of the READ and READ2 subroutines. The data format is three integer fields ending in Columns 5, 10 and 15 and one E15.5 field in READ and two E15.5 fields in READ2.

The active portion of the subroutine begins by reading a card and checking the first integer field for a function number. If a zero or negative number is found, control is returned to the main program. At the end of the data for each function a -1 in the first integer field signifies the end of that function. Control is then transferred to the initial read statement and the next function number is sought. Again, if a zero or negative value is found in the first integer field, control is returned to the main program. If a zero or -1 card immediately follows the -1 card at the end of the temperature-capacitor data, the program prints out an appropriate error message since Functions 8 and 9 must be used for each problem. If two successive -1 cards or a -1 card followed by a zero card is found within the function blocks, then control is returned to the main program. A flow chart of this subroutine appears in Appendix A.

Function 1 accepts data on cathode follower nodes. The first card of this data contains a 1 in the I1 field. The I2 field contains the number of the following node and the I3 field contains the number of the leading node on this and every succeeding card. A total of 25 cathode follower nodes are permitted. Calculations for this function begin at Statement 321 of the main program. In these calculations, the temperature of the following node is set equal to that of the leading node before each time step in the transient. The last card in Function 1, as in all other functions, contains only a -1 in the I1 field.

Function 2 accepts data on as many as 250 conductors for which radiation is the mode of heat transfer. The first card contains a 2 in the I1 field. The I3 field contains the conductor number and the first E field contains the value of K as defined on page 20 of Reference 1. The value of the radiation conductor is calculated at the start of each time increment starting at Statement No. 301 of the main program. An additional option is available in Function 2. If a minus is inserted before the conductor number in the I3 field, the subroutine will read the next card and assign the K value of that card to all conductor numbers beginning with the negative number and ending with the number on the card just read. Thus, if -49 and 59 appear in the I3 field of two successive cards, Conductors 49 through 59 will be assigned the K value that appears on the 59 card. Then if a thermal model contains a number of radiation conductors with the same K value, and the user numbers them consecutively, all

but the first and last cards may be omitted. The last card must contain only a -1 in the I1 field.

Function 3 accepts up to 50 pieces of data on those nodes for which latent heat data must be considered. All data are input in sets of two cards. The first card contains a 3 in the I1 field. This is also the first card of the first set of two. The I3 field of the first card contains the node for which the latent heat information is pertinent. The first E field contains the temperature at which the solid-liquid phase change occurs and the second E field contains the total amount of heat required for the node to undergo this change of phase. The second card of the set contains information only in the two E fields. This is the temperature and total heat required by the node to undergo the liquid-vapor phase change. The latent heat information is calculated before each time increment starting at Statement No. 326 of the main program. The last card contains only a -1 in the I1 field.

Function 4 accepts as many as 100 pieces of data on those conductors for which variable convection is the mode of heat transfer (constant convection conductors are input in subroutine TRCIN). The first card contains a 4 in the I1 field. This and all succeeding cards contain the conductor number in the I3 field, the exponent used in Equation 5-4 of Reference 1 is contained in the first E field and the area given in that equation is contained in the second E field. The variable convection conductors are calculated at the start of each time increment, beginning with Statement No. 306 of the main program. The last card contains only a -1 in the I1 field.

Function 5 accepts data for the reactor heating model currently in the program. The first card contains a 5 in the I1 field and the value of Q_0 as used in Equations 6-12 and 6-13 of Reference 1, is contained in the first E field. As many as 100 succeeding cards contain each node number in the I3 field and the power factor of the node (as described in the above-mentioned equations) in the first E field. The card following this data contains, in the I1 field, the number of the table where reactivity change vs time is given. The heat input from these nodes is calculated prior to the temperature calculations of each time increment at Statement No. 235 of the main program. The last card contains only a -1 in the I1 field. It should also be noted that Function 13 must be used whenever Function 5 is used.

Function 6 accepts data on the aerodynamic heating functions. The first card contains a 6 in the I1 field, a +1 in the I2 field if the Ramo-Wooldridge method is to be used (blank or zero if the Eckert method is used), a +1 in the I2 field if B is not 1.0 as described on page 46ff of Reference 1. Successive information appears on sets of two cards; up to 25 sets may be provided. The first card contains the node number in the I3 field and the value of K from Equation 6-16 of Reference 1 is contained in the first E field. The second card is used only if B is not 1.0. The I3 field contains a +1 if the node is a top surface (a zero or blank is used if it is a bottom surface), and the first E field contains the value of α , the local wing subtended angle. The last card contains only a -1 in the I1 field. The calculations for aerodynamic heating begin at Statement No. 311 of the main program. It should be noted that these calculations require input in several tables and also some input into Function 9.

Function 7 accepts up to 250 pieces of data which identify the change of one variable as a function of another. The first card contains only a 7 in the I1 field. Succeeding cards contain the dependency information. The I1 field contains the number of the dependent variable, the I2 field contains the number of the independent variable, the I3 field contains the table number in which the dependency is specified and the first E field contains the multiplying factor of the resultant if it is different than 1.0. The last card contains only a -1 in the I1 field. The arbitrary function interpolations are calculated at the start of each time increment beginning at Statement No. 291 of the main program.

Function 8 accepts as many as 700 pieces of data on the print specifications. It is possible to print out the value of any conductor or resistor number or the temperature, thermal capacitance or heat input to any node. The first card contains an 8 in the I1 field. The I2 field contains the class of variable to be printed and the I3 field gives the number of the variable to be printed. An option is also included whereby only the first and last cards of a consecutive sequence of one output variable need be included. The variable number on the first card is the negative of its true value. Thus, for Variable Class 2 (temperature), to print nodes 1 through 100 only two cards need be included; a 2 in the I2 field of both cards and -1 in the first card and 100

in the second in the I3 field. The last card contains only a -1 in the I1 field. The print specifications are used beginning at Statement 400 in the main program.

Function 9 accepts data on a series of problem constants used at various locations in the main program. The first card contains only a 9 in the I1 field. The problem constant number is given in the I3 field and information pertaining to the problem constant is given in the two E fields. Constant 1 establishes the print interval and is used starting at Statement No. 231 in the main program. The print interval may be altered during the course of the transient. If the same print interval is desired during the transient, the 1 in Column 15 is used with the print interval specified in the first E field. If the print interval is to be varied, the time at which the print interval is to change is entered in the second E field. Up to 5 different print intervals may be specified in this way. Constant 2 is used to over-ride the time interval. A 2 is written in Column 15 and the two E fields are specified in the same manner as Problem Constant 1. If only one over-ride time interval is to be used throughout the transient, then only one card is used with a 2 in Column 15 and the interval given in the first E field. Problem Constant 2 is used, starting at Statement No. 223 in the main program. Program Constant 3, given in the first E field, is a multiplying factor with a magnitude less than 1.0. The factor modifies the time interval between iterations, to guarantee more precise answers within the stability criterion. It is used in Statement 227 of the main program. Problem Constant 4 is the problem completion time and is used in Statement No. 335 of the main program. Problem Constants 5-8 are used in the aerodynamic heating calculations previously described. Problem Constant 9 is the initial time of the transient and is used in Statement 200 of the main program. Problem Constant 10 is the minimum allowed time interval. It is used in Statement 228 of the main program. The last card in Function 9 contains only a -1 in the I1 field.

Function 10 accepts data on heat transfer by incompressible fluid flow. The first of as many as 250 cards contains only a 10 in the I1 field. The input on the following cards contains the table number, where w_c vs time is found, in the I1 field, the following node number in the I2 field, the leading node number in the I3 field, the w_c of the following node in the first E field, and the time at which the following node becomes filled with the incompressible fluid

in the second E field. Flow in Function 10 and other functions is from the leading to the following node. The time at which a node becomes full is usually zero and can be left blank if desired. The existence of this quantity permits more accurate solution of the problem where fluid is flowing into a previously empty region. If this number is other than zero the thermal capacitance of that node should be specified in the capacitance input block as a 0 or -1 node. The last card in the Function 10 input contains only a -1 in that I1 field. The option to use only the first and last cards of a sequence of nodes is also available in Function 10. A minus is placed before the leading node (the I3 field) of the first card. The program automatically sequences both the leading and following nodes, keeping all other information constant, through the node numbers found on the card representing the end of the sequence. Function 10 is used in the main program at Statement No. 230.

Function 11 accepts up to 50 data cards which permit the merging of two or more fluid streams into one node. The first card contains only 11 in the I1 field. Succeeding cards contain the data. In the I2 field is the node number into which the fluid is flowing. The I3 field contains the node number of one of the nodes being mixed and the I1 field contains the table in which the w_{cp} of that node is described as a function of time. The last card contains only a -1 in the I1 field. Function 11 is used immediately following the completion of all calculations of temperature change during each time increment at Statement No. 280 in the main program.

Function 12 accepts data which permits the passage of fluid through a series of nodes in one time interval. The input in Columns 1-15 is identical to that of Function 10. The first E field contains the solid portion of the node capacitance. Function 12 also permits entry of the first and last cards of a sequence by inserting a minus before the leading node of the first card. The leading and lagging nodes are sequenced and all other information is kept constant in a manner identical to the use of Function 10. The last card in Function 12 contains only a -1 in the I1 field. Function 12 is used at Statement No. 266 in the main program.

Function 13 accepts data on the reactivity temperature feedback to the reactor calculations as well as data on the use of a deadband controller. The

first card contains only I3 in the I1 field. The next three cards contain the node number and the reactivity coefficient of the three nodes which it has been assumed adequately describe the temperature feedback to the reactor. If no deadband controller is used, the next card contains a -1 in the I1 field. If a deadband controller is used, the next card contains the node to which the deadband characteristics are applied in the I3 field, the reactivity insertion increment is given in the second E field. The second deadband card contains the temperature to which the critical node is being compared in the first E field and the deadband around this temperature which must be exceeded before a reactivity insertion is applied in the second E field. These quantities are used in the reactor heating calculations (Function 5). The last card contains a -1 in the I1 field.

X Function 14 accepts data to permit the early completion of a run if steady state has been reached. The first card contains 14 in the I1 field, and the relative error per time increment which is considered sufficient to determine steady state in the first E field. It is also possible that a series of steady-state conditions could exist. For each succeeding steady-state condition, a card is added. The I1 field contains the card number (up to 20 may be used) and the first E field contains the time to which the problem will automatically be advanced. The last card contains -1 in the I1 field.

3.4 SUBROUTINE TABIN

Subroutine TABIN is called from the main program immediately following the call to FIN. Its purpose is to read in the data presented to the program in the form of tables. The subroutine calls the READ2 subroutine to read the data as presented in three I fields and two E fields. COMMON VARY is used to collect and pass data on those conductors which vary during the course of a transient. COMMON FUN7 serves the same purpose since the only way conductors could be varied in the form of tables is through the use of Function 7 and its COMMON FUN7.

The subroutine begins by reading a card containing the table number in the I1 field, the class of independent variable in the I2 field, the class of dependent variable in the I3 field, the value of the independent variable in the first E field and the value of the dependent variable in the second E field. A check is made to see if the dependent variable is Type 1. If this is true,

then the appropriate conductor number is obtained from the data in FUN7 and is stored in VARY so that only calculations on these variable conductors need be performed.

Successive cards are now read and these contain the independent variable in the first E field and the dependent variable in the second E field. These data (up to a total of 52 cards including the first) are read in until the graph of the function has been described in tabular form. All tabular data are later searched by a linear interpolation routine when used in the program calculations. The last card at the end of each table must contain only a -1 in the I1 field.

3.5 SUBROUTINE READ

Subroutine READ is used for the purpose of reading and then printing the input to the program which has a format of three I fields and one E field. After this information is read, it is passed back to the calling program through the arguments in the calling sequence. This subroutine is called from the conductor, resistor input section of TRCIN and from many locations in FIN. The first column could contain the letter C if used in subroutine TRCIN, to designate that the value in the E field is a conductance rather than a resistance. If a C is found, then KFLAG is set to zero and passed back to TRCIN through COMMON FLAG for appropriate notation. KFLAG is not used in subroutine FIN. The format contains 50 columns of Hollerith information to permit the user to include comments on his input cards.

Entry READ2 is used for the purpose of reading and then printing the input to the program which has a format of three I fields and two E fields. The subroutine is called from the temperature-capacitor input section of TRCIN, from several locations in FIN and from TABIN. The information is read from cards and passed back to the calling program through the arguments in the calling sequence. The format contains 35 columns of Hollerith information to permit the user to include comments on his input data.

3.6 SUBROUTINE FLOW

Subroutine flow is divided into two parts. The first part is called from the main program at Statement No. 229. Its purpose is to account for the transport of heat by incompressible fluid flow when the heat transfer coefficient to the adjacent walls is well known. Input to this section of the subroutine is through Function 10 and any tables specified in that function. This information is carried in COMMON FUN10. The second part of the subroutine begins at entry FSTFLO and is called from Statement No. 265 in the main program. Its purpose is the same as for the first part of the subroutine, except that the fluid moves through several nodes in one time increment (DELTAT). Input to this section of the subroutine is through Function 12 and any tables specified in that function. This information is carried in COMMON FUN12.

The first part of the subroutine begins by initializing error flags. Then the current w_{c_p} from each of the tables referenced in Function 10 is calculated. This calculation is performed by a linear interpolation using function subprogram ENTERP. The ratio of the minimum node capacitance to the current w_{c_p} value is calculated for each table referenced. The units of this ratio are time. If this ratio is less than DELTAT, then DELTAT is adjusted to be equal to the ratio. A discussion of this point can be found in Section 6.4 of Reference 1. At this point, the calculations are performed to find the temperature of the following nodes indicated in the Function 10 input. If a Type II (positive capacitance) node is indicated, the temperature of the fluid remaining in the node and the fluid entering are averaged. This temperature is then used in the temperature calculations in the main program as indicated in Equation 2.7.4. If the node is a Type I (zero capacitance) node, the following node is set equal to the leading node. Calculations are then later performed according to Equation 2.7.5.

Another test made in this section is whether or not to perform the flow calculations on any given node. The Function 10 input permits calculations on any fluid node to be omitted until a specified time in the transient. This permits simulation of the condition where a fluid node is initially empty and has fluid introduced during the course of the transient. Such a case would exist for the initial transient when a boiler is first filled or other similar circumstance.

A stability check is also made in this subroutine following Statement No. 30. If both the RC product of any of the flow nodes is less than the minimum RC product for the remainder of the nodal network and DELTAT is less than 1 for two successive iteration, this is considered a fatal error and the calculations are terminated. If the stated conditions were present, then on each successive iteration the DELTAT would decrease until it causes an under-flow condition in the computer core. This condition can only exist on a Type I node.

The second part of the subroutine begins at ENTRY FSTFLO. On the first pass through the subroutine, the temperature of each node from the previous time step (TEMP) is initialized. In succeeding passes, the program initializes error flags unique to this section of programming. First, the w_{cp} referenced in each of the the tables named in Function l2, is calculated. Then the capacitance of each of the following nodes is calculated according to Equation 2.7.2. The stability of each node is checked according to the same criteria used in the first part of the subroutine. If the node passes the stability criteria, the temperature of each node is calculated according to Equation 2.7.7.

3.7 SUBROUTINE ENTERP

Function subprogram ENTERP is a linear interpolation routine and is called from a number of locations throughout the program. It searches the data within the table being referenced and performs the requested interpolation. The table number is available through common TABNO. The arguments to the subprogram are the argument (ARG) and the value of the first number in the table of interest (TAB).

A check is first made to see if the table has a constant dependent variable. This is denoted by a non-zero value in the second position of the table. In this case the constant is passed back to the calling routine as the result of the function calculations.

If this is not true, the subroutine proceeds to a search of the table to calculate the result. If the independent variable in the table increases with higher table locations (as will be true for most cases), then the search proceeds until the independent variable exceeds or is equal to the argument. If it is equal, then the corresponding dependent variable is passed back as

the result. Otherwise, a linear interpolation, between the points that bracket the argument, is performed.

If the table describes a periodic function, then the first location in the table gives the period. Periodicity is tested and appropriate modifications to the argument made if the test is positive.

If at any time it is found that the argument is less than the lowest independent variable in the table or greater than the highest independent variable, then an error message is printed out giving the argument and the table number in which the search was taking place. This is a fatal error condition and results in an immediate termination of the job.

3.8 SUBROUTINE TAVG

Subroutine TAVG is called from Statement No. 281 in the main program. It is used to average the temperature of two or more fluid nodes which join to form a common node. Input to this subroutine is through Function 11 with the data being stored in COMMON FUN11. The subroutine first sets the sum of the capacitances (CSUM) and the sum of the capacitance-temperature product (CTSUM) to zero. An error flag is initialized before the calculations proceed.

The calculations are performed according to Equation 2.7.6. To perform the calculations the inlet node (INNOD) and the joining node (NUNOD) are noted along with any table which carries data of w_c vs time. When NUNOD is the same as that of the previous pass through the DO loop, then CSUM and CTSUM are increased by the appropriate amount. If NUNOD is changed from the previous pass, then the capacitance and temperature of node NUNOD are calculated. A stability check is made, as in subroutine FLOW, and if the problem is unstable an error message is printed and the solution process stopped. It is necessary for the stability check to fail on two successive passes through the subroutine before the error is noted. After these calculations have been completed, the next joining node is noted and further calculations performed until all the Function 11 data have been analyzed. Control is then passed back to the main program.

3.9 SUBROUTINE HEATIN

Subroutine HEATIN is called from the main program at Statement No. 235. Its purpose is to perform nuclear heating calculations using a simplified reactivity model as the driving force for thermal calculations. Input for this subroutine is through Functions 5 and 13. This nuclear heating model has been added to satisfy the requirements of SNAP-8 thermal transient analysis. It would be possible to replace this heating model simply by replacing this subroutine and the arguments to it, by altering Functions' 5 and 13 inputs, and by altering COMMON FUN5 and FUN13 in the main program, in subroutine FIN and in this subroutine.

The subroutine starts by calculating the predetermined change in reactivity if any exists. This is done by interrogating the table specified in the Function 5 input, which gives reactivity input as a function of time.

A check is next made to determine whether a deadband controller is used as part of the analytical model. If it is not, control is passed to Statement No. 10 in this subroutine. Otherwise, a check is made to see if the temperature of the node specified as NDEADB exceeds the base deadband control temperature by more than the allowable deadband limit. If it does not, the heating calculations are continued. If the temperature of node NDEADB is outside the deadband limits, a check is made to determine if the reactivity change permitted by the deadband controller has been made within a time period specified in the input to Function 13 as DBWAIT. If not the proper reactivity adjustment is made. If the last controller change has been too recent the heating calculations continue at Statement No. 10.

At Statement No. 10, the first calculation is the total reactivity change driving the reactor. This is the sum of temperature reactivity feedbacks from three lumped nodes, the deadband contribution and the predetermined reactivity change. The calculations proceed according to Equation 2.7.8. (This assumes that the reactor is nearly critical during the calculations.) Should the reactor be in a prompt critical state, the calculations are stopped by branching to Statement No. 910.

The exponents used in the referenced equation are next calculated and tests are made on these numbers to avoid overflows in the computer. A DO loop

is then entered to calculate the heat generated in each node. It is assumed that the basic reactor operation is at some power level Q_0 . The beta and gamma heating is some fraction of this power level. It is further assumed that beta and gamma heating need be considered only in the case of a shutdown and that if a shutdown occurs it will start at time zero. Thus, if the power is any fraction greater than 6% of its original level, then 6% of the power is assumed to be from the decaying beta and gamma heat. When calculations have been completed for each of the nodes specified in the Function 5 input, control is returned to the main program.

3.10 SUBROUTINE STEADY

Subroutine STEADY is controlled by input from Function 14 and is called following Statement No. 270 in the main program. Its purpose is to terminate the execution of the program once equilibrium conditions occur without waiting until the problem completion time specified in Function 9 has been reached. Use of this subroutine by choosing Function 14 can present a significant monetary saving since it is often quite difficult to accurately estimate the time at which equilibrium is reached. An overestimation results in the use of excess computer time. An underestimation means wasted time because the computer run must be resubmitted.

The error criterion on which the assumption of equilibrium is based is fractional change in temperature per unit of time. The temperature of each node (T), the maximum number of nodes (MAXNO), and the time increment (DELTAT) are passed to the subroutine as arguments. The argument KEY is determined in the subroutine and passed back to the main program. If its value is 1, then equilibrium has been attained.

The subroutine first sets KEY to 1 assuming that equilibrium will have been reached. It then compares the fractional change in the temperature of each node in the latest time step (DELTA) with the error criterion given. If DELTA is greater than the error criterion (ERROR) then KEY is set to zero and the search ended for this time step. If DELTA is less than ERROR, the search continues through the remaining nodes. Before control is returned to the main program, the temperature of each node is saved as TSAVE to be used on the next iteration to determine the fractional temperature change in this subroutine.

3.11 SUBROUTINE LAT

Subroutine LAT is called from Statement No. 327 in the main program. Its purpose is to correctly follow the temperature of any node which could go through a phase change. Temperatures of these nodes are calculated normally in the program and then modified as necessary in this subroutine. The modification occurs when a node crosses a phase change temperature boundary. When this happens the subroutine keeps the nodal temperature at the phase change temperature until sufficient heat has been added to or taken from the node for it to again change temperature in its new phase. It should be noted that any node which can undergo a phase change must also have its thermal capacitance specified as a function of temperature in Function 7. Input to the latent heat subroutine is through Function 3 in subroutine FIN.

The subroutine is entered separately for each node which could undergo a phase change. The determination is first made whether the node is currently undergoing a phase transition ($QDEL=0$). If not and if neither the current temperature nor the temperature during the previous iteration was at a phase transition temperature, then the current temperature is noted as $TPREV$ so that a similar comparison with the previous temperature can be made on the next iteration. If a phase change is taking place, the change in heat to the node during the current iteration period is noted. If it is sufficient to change the node from the transition temperature then $QDEL$ is reset to zero and the node is again permitted to undergo temperature changes. During each pass through the subroutine both the upper and lower phase transition temperatures are checked ($PHT1$ and $PHT2$, respectively).

3.12 SUBROUTINE QECK

Function subprogram QECK is called immediately following Statement No. 311 in the main program. Input for this subroutine is through Function 6 and through some of the problem constants specified in Function 9. The purpose of the subroutine is to calculate aerodynamic heating using the Eckert equation (Equation 2.7.13).

Most of the information used in the equation is contained in tables as noted in the description of subroutine FIN. The flow path through the subroutine is simply one of interpolating in each table using function ENTERP. If

errors are encountered in any of the interpolations, an error flag is set and returned to the main program. At that point the work on the problem is stopped and data for another problem would be read in if it exists. After the proper values have been interpolated from each of the tables they are used to calculate QECK and this value is returned to the main program as heat input for the node specified.

3.13 SUBROUTINE QRW

Function subprogram QRW is called immediately following Statement No. 316 in the main program. It is used to calculate aerodynamic heating using the Ramo-Wooldridge equation (Equation 2.7.12). Input is through Functions 6 and 9, just as for function QECK.

Calculations proceed in the same manner as for function QECK. The information used in the Ramo-Wooldridge equation is obtained by interpolating in the tables described in subroutine FIN. If errors are encountered calculations on the problem are stopped. The appropriate values are then used to calculate QRW and this value is returned to the main program as heat input for the node specified.

4.0 FORTTRAN VARIABLES AND THEIR USE IN THE PROGRAM

<u>Variable</u>	<u>Description</u>
C	Thermal capacitance of the subscripted node
Q	Heat input to the subscripted node
T	Temperature of the subscripted node
CO	Problem constants specified in Function 9 of the input data
FD	Floating point data in the first floating point field of the input cards (Col. 16-30)
I1, I2, and I3	Integer data in the first, second, or third integer field of the input cards (Col. 1-5, 6-10, and 11-15, respectively)
KC	Node specified as a cathode follower in Function 1
KR	Conductor specified as having radiant heat transfer in Function 2
QO	The base heat output of the reactor, used in the nuclear heating routine
RC	The RC product of the subscripted node; calculated as the capacitance of the node divided by the sum of the conductances surrounding the node
RI	The sum of the conductances of the subscripted node
TR	The temperature of the subscripted node divided by the sum of the conductances
ADD	Multiplier of the print frequency which determines the time at which output is printed
AER	The value of K in Equation 6-16 of Reference 1
ARG	The independent variable passed to function ENTERP as the argument
EKR	The coefficient K_{ij} used in radiant heating calculations as described in Equation 5-5 of Reference 1 ($K_{ij} = \sigma F A_{ij}$)
Fl4	A flag set to 1 if function 14 is to be used
IKF	A counter to determine the number of pieces of data used in Function 7
IPH	The node for which latent heat calculations are to be performed

<u>Variable</u>	<u>Description</u>
IPK	A counter to determine the number of pieces of information to be printed for each printout
I2V	Node number in the I2 field to which a conductor is attached
I3V	Node number in the I3 field to which a conductor is attached
KEY	A flag set to 1 if steady state has been successfully achieved
MIX	A counter to determine the number of pieces of data used in Function 12
NEW	The following node as designated in the fluid flow input
QCH	The total heat accumulated in a node which is undergoing phase transition
RHO	The temperature coefficient of reactivity of the node specified in the I3 field in Function 13
TAB	The table passed to function ENTERP for evaluation
TYM	The time at which the program $\Delta\theta$ currently being used is no longer used
TYP	The card field on the first data card of a run which must read "TITLE"
WCP	The WC_p of the subscripted table
ACON	The area for the subscripted convective heating conductor
COND	The conductance of the subscripted conductor
CSUM	The sum of the fluid capacitances flowing into a single node; used in subroutine TAVG
DBG0	The time at which the dead-band reactivity controller is again permitted to be moved
DELK	The δk of the reactor in the nuclear heating calculations
IAER	The node for which the aerodynamic heating calculations are being performed
IAF2	The node or conductor number of the dependent variable in the arbitrary function input
IAF3	The node or conductor number of the independent variable in the arbitrary function input

<u>Variable</u>	<u>Description</u>
IAF ⁴	The table number in which the arbitrary function is described
ICON	The conductor number described in the free convection function input
IERR	An error indicator set to 1 if either subroutine FLOW or TAVG have a mathematical instability
IFUN	An indicator set equal to various function numbers to indicate an error in that function
IPHK	A counter to indicate the number of pieces of data used in Function 3
IPR1	Specification of the type of output desired for the node or conductor number on that input card
IPR2	The node or conductor number for which output is desired
IRAD	A counter to indicate the number of pieces of data used in Function 2
IRB1	A counter to indicate the number of pieces of data used in Function 5
ITAB	The table number and classes of variables described in the table input
KDEP	The dependent variable type referenced when referencing a particular table
KIND	The independent variable type referenced when referencing a particular table
KNEW	The node number of the leading node as used in subroutine FLOW
LNEW	The following node number in the input for the fast fluid flow
NODQ	The node number referenced in the nuclear heating calculations (subroutine HEATIN)
PHH1	The amount of heat necessary for the referenced node to pass through the solid-liquid transformation
PHT1	The temperature at which the solid-liquid phase transformation occurs for the referenced node
PHH2	The amount of heat necessary for the referenced node to pass through the liquid-vapor transformation
PHT2	The temperature at which the liquid-vapor phase transformation occurs for the referenced node
PRNT	The time at which the next output is to be printed

<u>Variable</u>	<u>Description</u>
QDEL	The amount of heat stored or lost by the referenced node during the time it was undergoing a phase change
TEMP	The temperature saved from the last iteration in the fast fluid flow calculations
TIME	The problem time (used in subroutine LAT)
TNEW	The temperature of the leading node as used in the fluid flow calculations
TOME	A flag set to zero at the start of the problem to denote the first iteration; on the first iteration some calculations are performed for utilizing values and not repeated, other calculations are skipped because these initializations have not yet taken place; after the initial iteration TOME is set to 1.
TZME	The problem time
ALLOC	The local wing subtended angle in the aerodynamic heating calculations
CFMIN	The minimum fluid capacitance of each table referenced in the Function 10 input
CTSUM	The sum of the capacitances times the temperatures of the fluid nodes being merged in subroutine TAVG
DBTOT	The total dead-band controller contribution to the reactivity change, as used in the nuclear heating calculations
DELKC	The predetermined reactivity change in the nuclear heating calculations as determined from the referenced table
DELTA	The calculations of the percentage temperature change per unit time in comparing whether steady state has been attained
ERROR	The percent rate of change per unit time of the most rapidly changing node which is used as the criterion of attaining a steady state condition
EXCON	Exponential constant used in the convection conductor calculations and input in Function 4
FIAF	A multiplying factor used in any of the arbitrary function designations
IAERK	A counter to indicate the number of pieces of data in Function 6
IAER1	A number in the I2 field of the aerodynamic heating input; if zero, the Ramo-Wooldridge calculations are performed - if one, the Eckert method is used

<u>Variable</u>	<u>Description</u>
IAER2	A number in the I3 field of the aerodynamic heating input; if zero, then β is a function of α - if one, the functional relationship does not exist
ICALC	A flag set in the main program to determine if temperature calculations are performed on all nodes during that iteration or only on some designated nodes
ICATH	A counter to indicate the number of pieces of data in Function 1
ICONK	A counter to indicate the number of pieces of data in Function 4
IDELK	Table number in which is described the predetermined reactivity change versus time in the nuclear heating calculations
INNOD	The node number of one of the fluid flow nodes being averaged in a common node (used in subroutine TAVG)
IPASS	A flag to denote the first or second pass through Section TRA of subroutine TRCIN during the iteration scheme
IRVAR	An indicator which is set equal to the conductor number if the conductor is variable; it permits recalculation of <u>only</u> variable conductors during the transient
ISTEP	A counter which helps point to the data on frequency of printed output
IZAER	A flag to determine if the aerodynamic heating is for a top surface (=0) or a bottom surface (=1)
JTABL	The table number in which is stored the $w_{c,p}$ versus time information for the fluid flow node used in the averaging scheme of subroutine TAVG
KEY10, KEY11, and KEY12	Flags set in the use of Functions 10, 11, or 12 which denote an instability in the use of the data from one of those functions
KLEAD	The leading node in the fluid flow calculations
LLEAD	The leading node in the fast fluid flow calculations
MAXNO	The maximum node number used in the problem
MINND	The node which has the minimum RC product
NBASE	A node which is used as a reference for the temperature feedback in the nuclear heating calculations

<u>Variable</u>	<u>Description</u>
NODEQ	One of the node numbers input for which nuclear heating calculations are performed
NSTEP	A counter which helps point to the data on time steps chosen for the iteration scheme
NUNOD	The node number into which two or more fluid nodes are being combined and their temperatures averaged
PRINT	Space reserved in the main program to permit printing of any temperature, capacitance, conductance, or heat flow, or a node or conductor by addressing the proper location within PRINT
PRTST	A cumulative counter used to test when in the transient printed output is requested
PRUNT	The area in which the numbers to be printed are stored before the print operation occurs
QBETA	The beta-gamma heating of the subscripted node as used in the nuclear heating calculations
RCMIN	In minimum RC product
RIFIX	The $1/R$ sum for the fixed conductors
TABLE	The location within the tables where a particular functional relationship is described
TBASE	The temperature of node NBASE at which reactivity feedback to the reactor is zero
TFLAG	A flag set to 1 after the first pass through Section TRA of subroutine TRCIN; if all the capacitors are non-zero, the flag remains set and the second pass through TRA is skipped
TPREV	The temperature of the node used in the latent heat calculations on the previous iteration
TSAVE	The temperature of the subscripted node on the previous iteration as used in the determination of the attainment of steady state
WCPDT	The current value of $w_{c,pdt}$ for the subscripted table
ADTIME	The time to which T2ME is advanced after a steady state condition has been reached
CFLUID	The capacitance of the subscripted following fluid flow node
CSØLID	That portion of a following fast fluid flow node which is due to the surrounding walls

<u>Variable</u>	<u>Description</u>
DEWAIT	The minimum time after the movement of the dead-band feedback controller before the controller can be moved again in the nuclear heating calculations
DELKDB	The reactivity change which occurs on each movement of the dead-band controller
DELTAT	The time step used in the current iteration
DELTDB	The dead-band within which a temperature change is permitted without causing activation of the dead-band controller (the limit is <u>+DELTDB</u>)
FULTIM	The time at which one of the fluid flow nodes becomes filled and calculations on the node commence
INNØDE	One of the inlet node numbers which flows into a common node to be averaged in subroutine TAVG
IRCVAR	A series of flags to denote those nodes which have either a variable capacitance or are tied to a variable capacitor; these are denoted by making IRCVAR of the node equal to the node numbers; Function 12 nodes are the negative of the node number
JOINMX	A counter to indicate the number of pieces of data in Function 11
JTABLE	The table number in which the w_{cp} versus time for one of the leading nodes for the fluid flow averaging calculations is described
KERR10, KERR11, and KERR12	A flag set to denote an instability in subroutines FLOW or TAVG; the flag is set after detection on one pass through the subroutine and calculations are stopped for detection on a second successive iteration; instability indicates an error in Functions 10, 11, or 12
KFOLOW	The number of the following node as calculated in the consecutive node input section of Function 10
KKTABL	The identification of the table in which the w_{cp} versus time information is stored for use during the fast fluid flow calculations
KPUNCH	A flag read in on the "TITLE" card; if set to 1, the program punches the temperature-capacitance cards at the completion of the run
KTABLE	The identification of the table in which the w_{cp} versus time information is stored for use during the fluid flow calculations
KTHMAX	A counter to indicate the number of pieces of data in Function 10

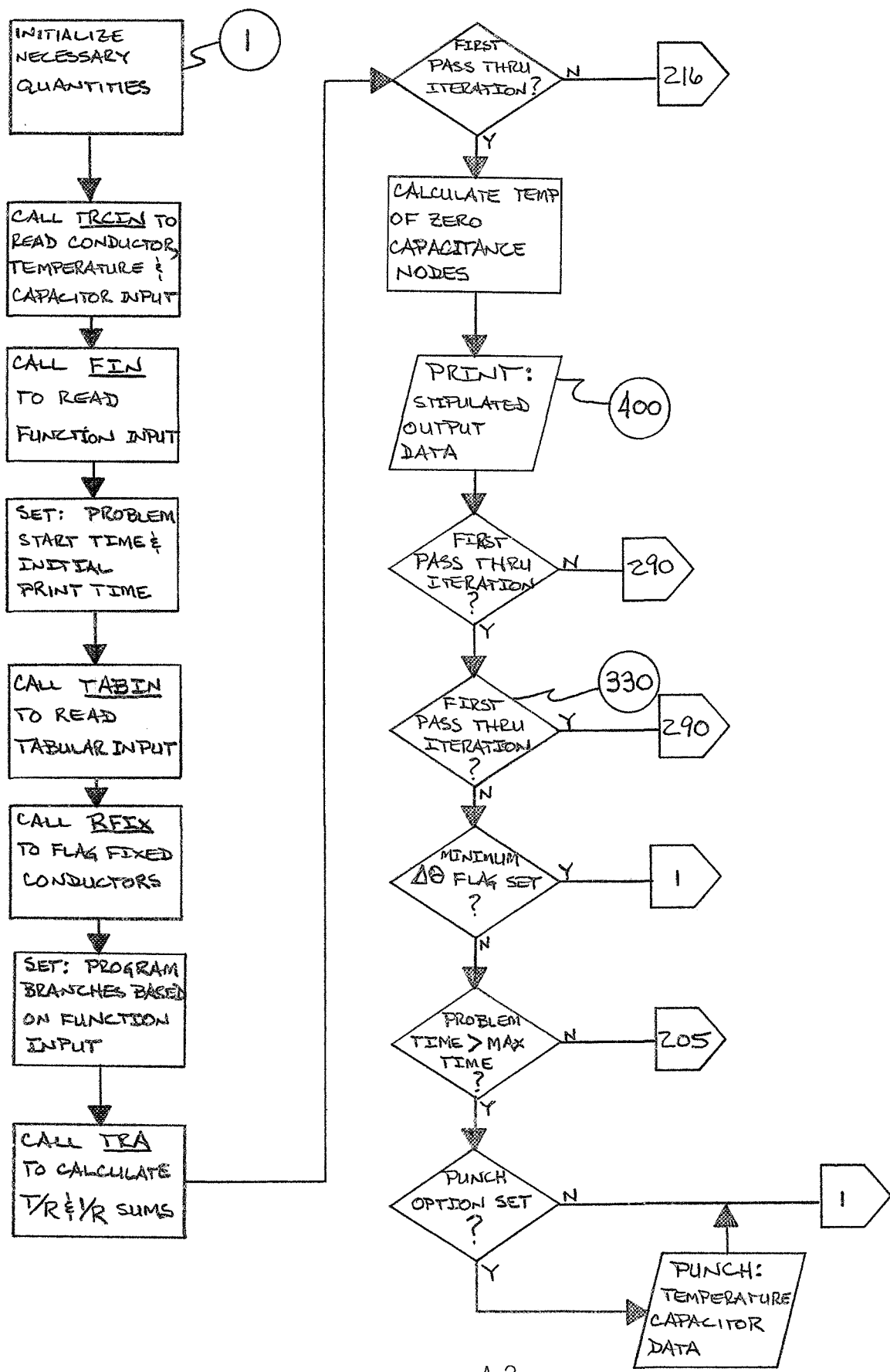
<u>Variable</u>	<u>Description</u>
KTHNEW	The node number of the following node used in the fluid calculations
LASTPR	A counter which indicates the number of printed outputs requested since the last time the print frequency changed
LFOLOW	The number of the following node as calculated in the consecutive node input section of Function 12
LLTABL	The table number where the w_{cp} versus time information, for the node referred to in the Function 12 input, is stored
LTABLE	The table number where the w_{cp} versus time information, for the node referred to in the Function 10 input, is stored
MAXNOR	The maximum conductor number used in the problem
MININT	A flag set to 1 if the time interval is less than the minimum specified as problem constant 10 of Function 9
NDEADB	The node number on which the dead-band controller action is based
NEWNOD	The number of the following node in the first card of the consecutive node input section of Functions 10 and 12
NODCLC	A reference area where the number of time intervals since the last temperature calculation, for each of the nodes, is stored
NODVAR	A flag which is set if any capacitors or conductors are variable
NUNODE	The junction node number into which two or more fluid nodes flow in subroutine TAVG
NUTIME	A flag which is set to the number of steady state cycles the problem will handle; when the number is exceeded the time is set past the problem completion time so that calculations stop
POWFAC	The power factor of the subscripted node to indicate the fraction of Q_0 from the nuclear heating calculations being generated in that node
PRTIME	The time at which the next printed output will occur
PRUNT2	The storage area containing K-, T-, C-, or Q-, depending on whether the printed output is a conductance, temperature, capacitance, or heat flow
SIGNAL	A logical record tested at various locations in the program to determine if a fatal error has occurred and the run should be aborted

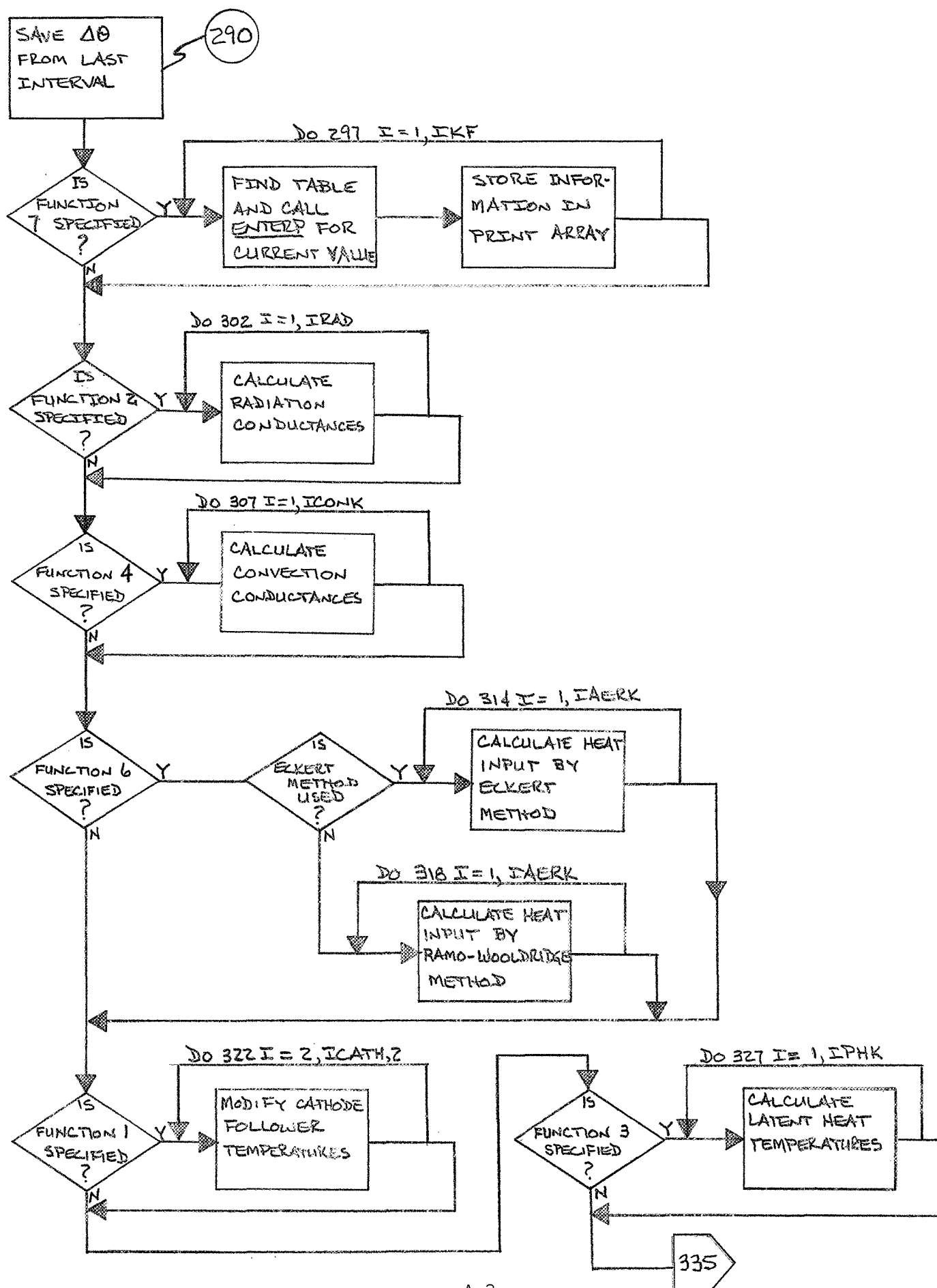
VariableDescription

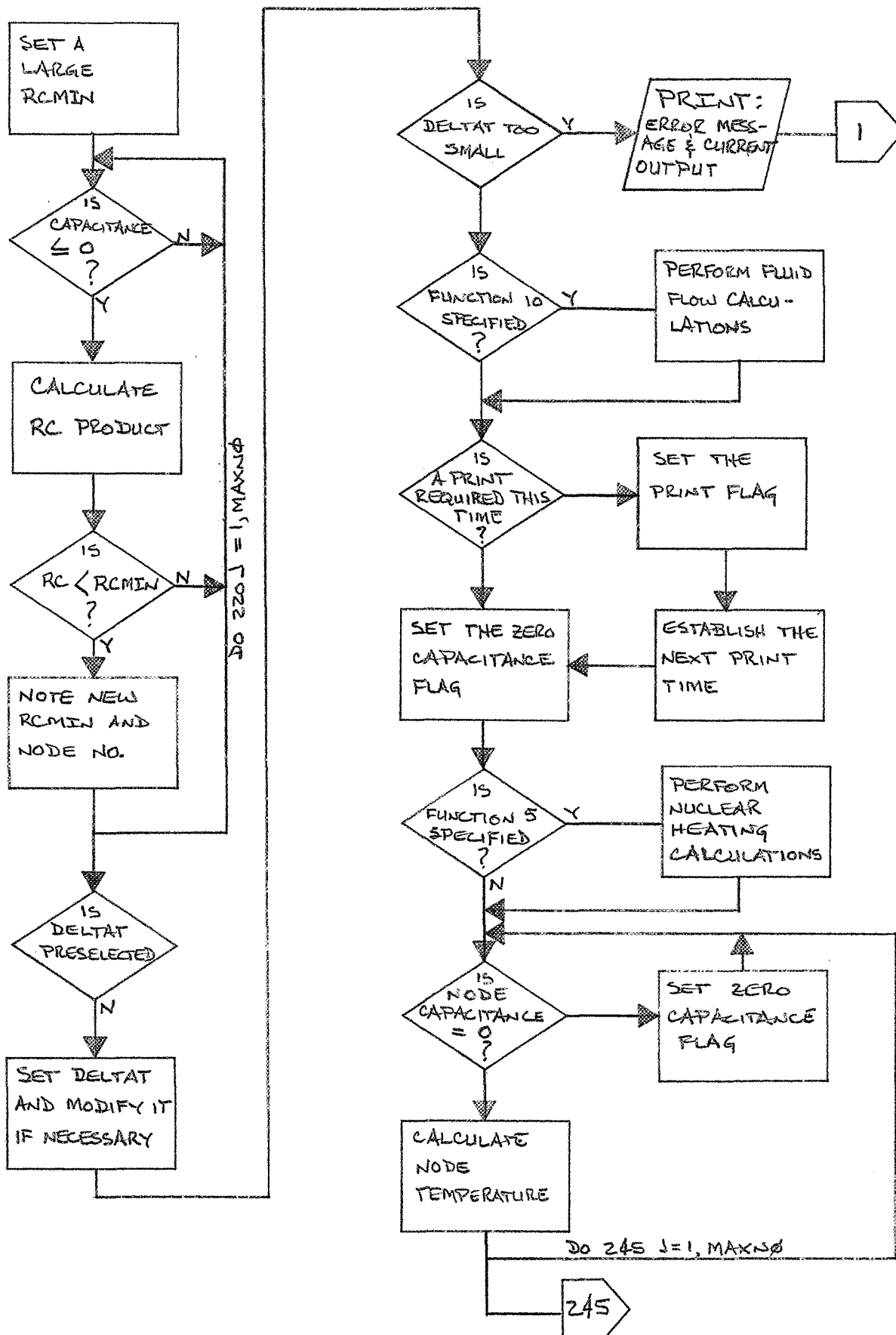
TDEADB	The temperature upon which the dead-band controller action is based
TIMMAX	The maximum time specified in Constant 2 of the Function 9 input during which the time interval is determined

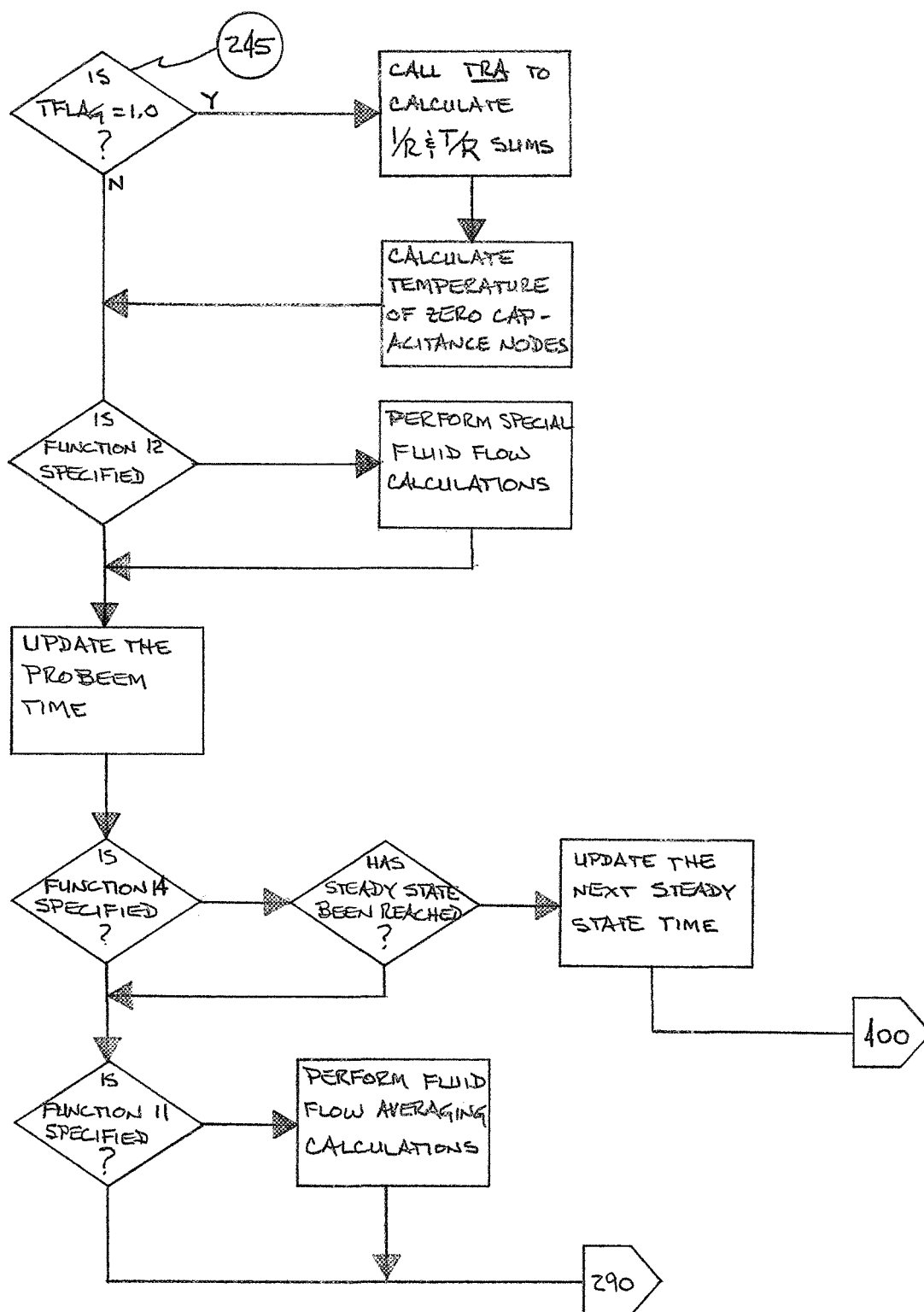
A P P E N D I X A

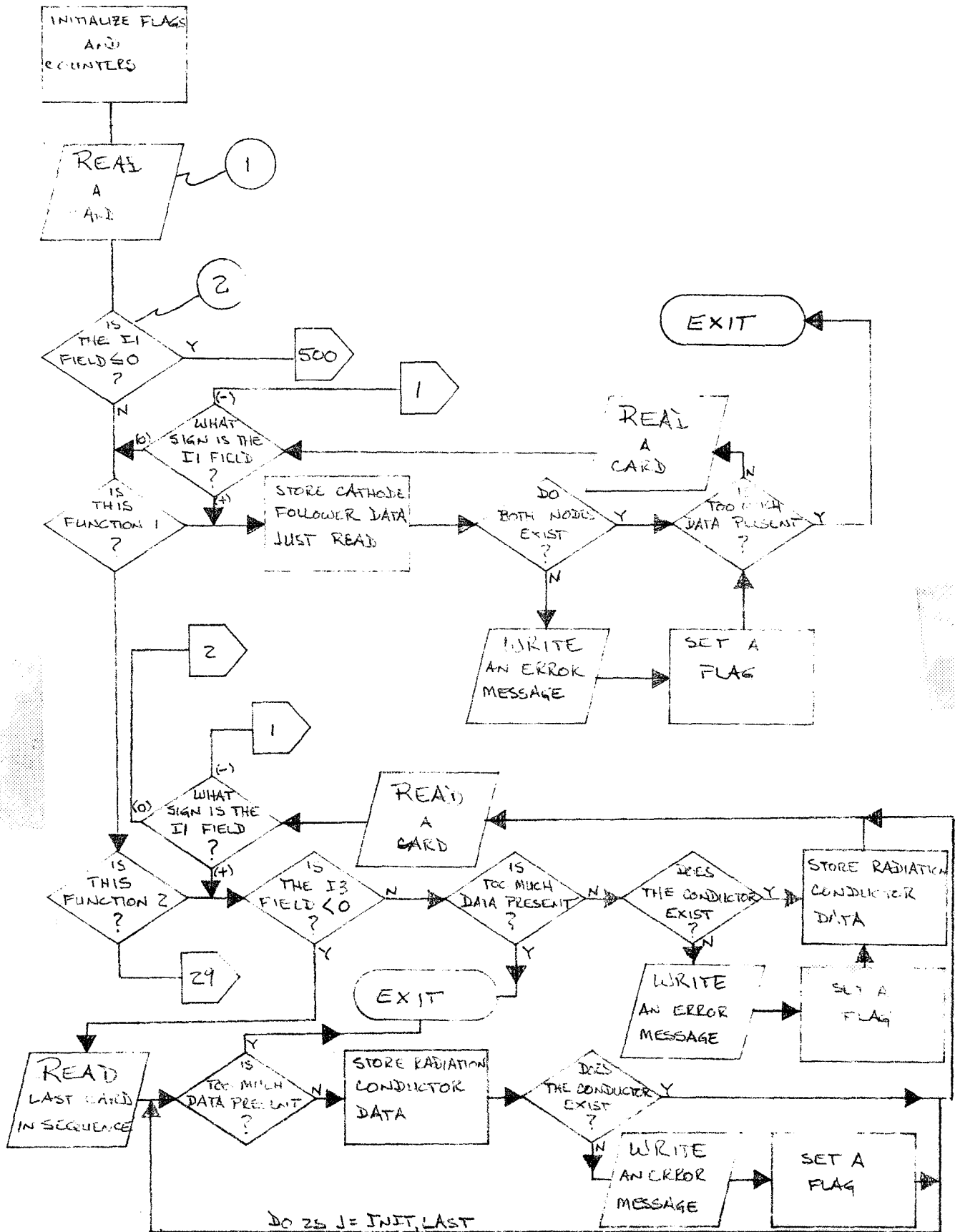
MAIN PROGRAM

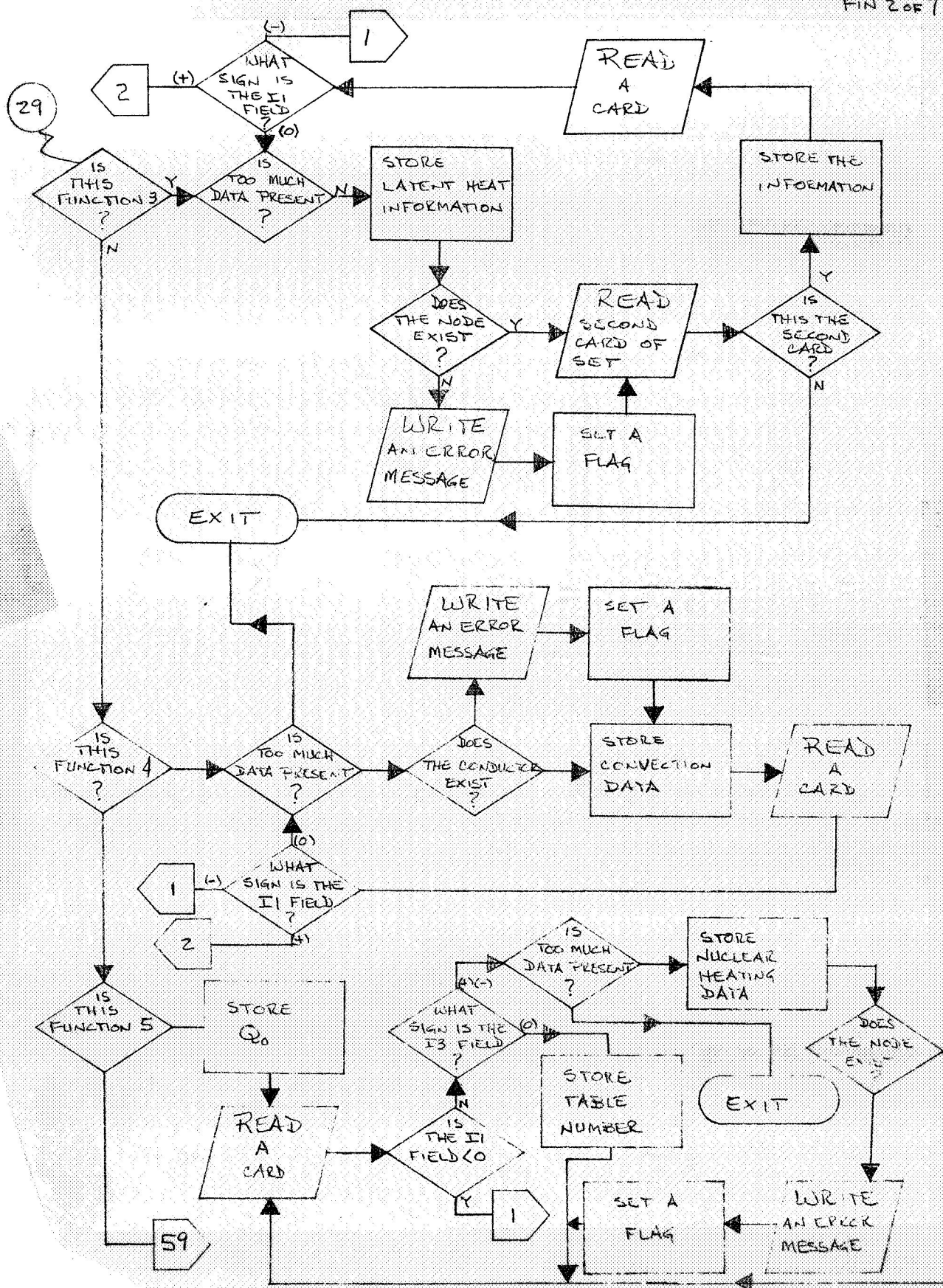


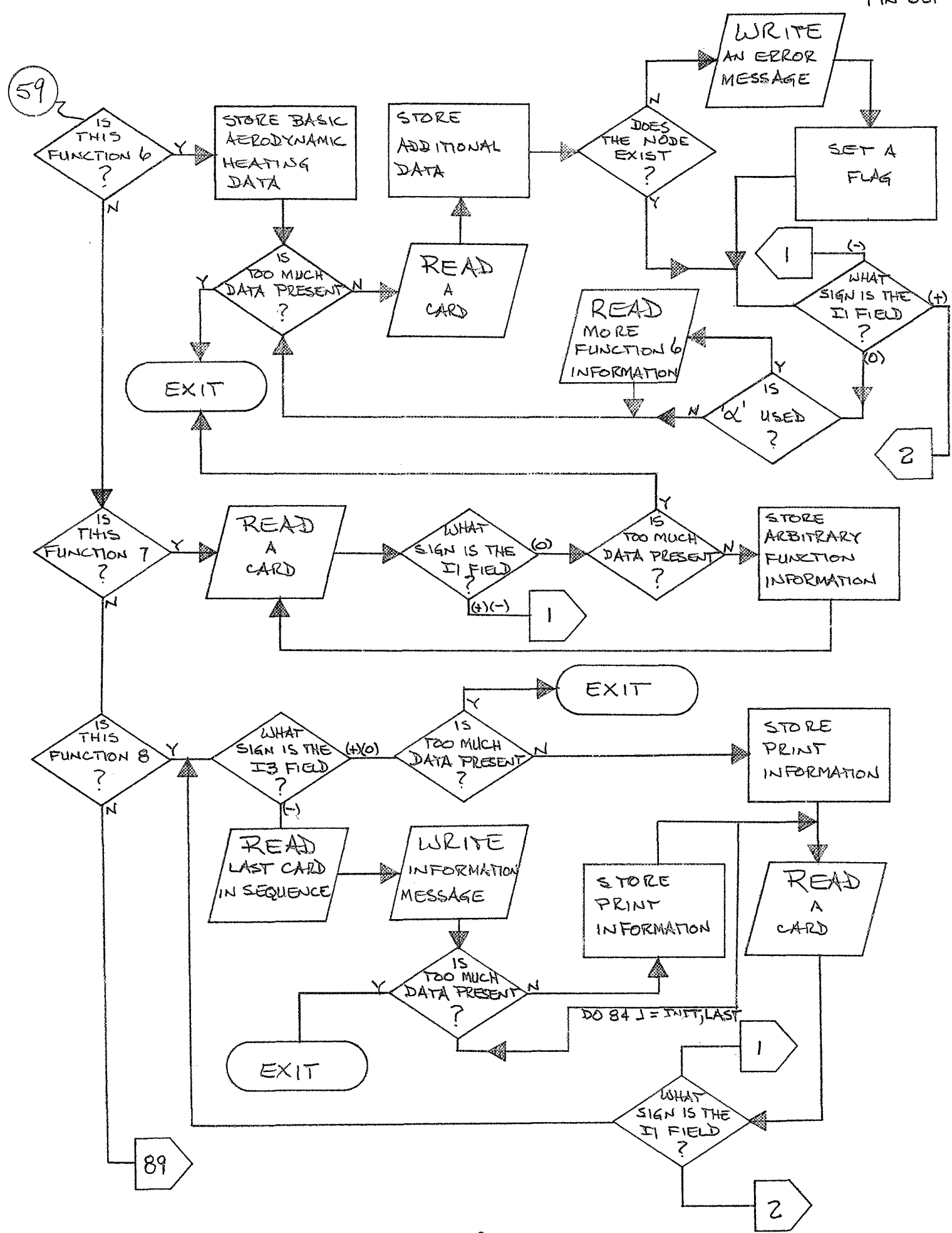


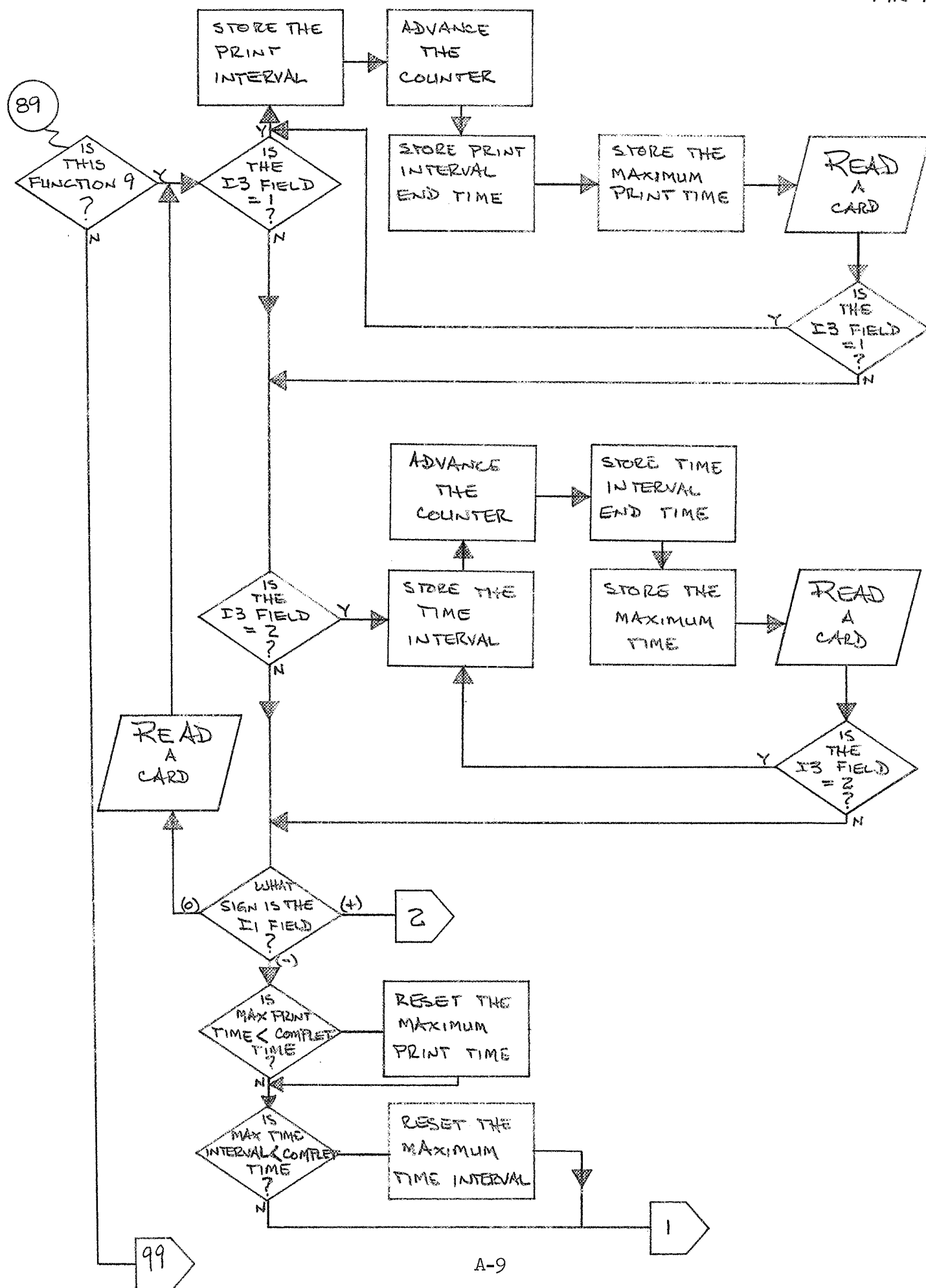


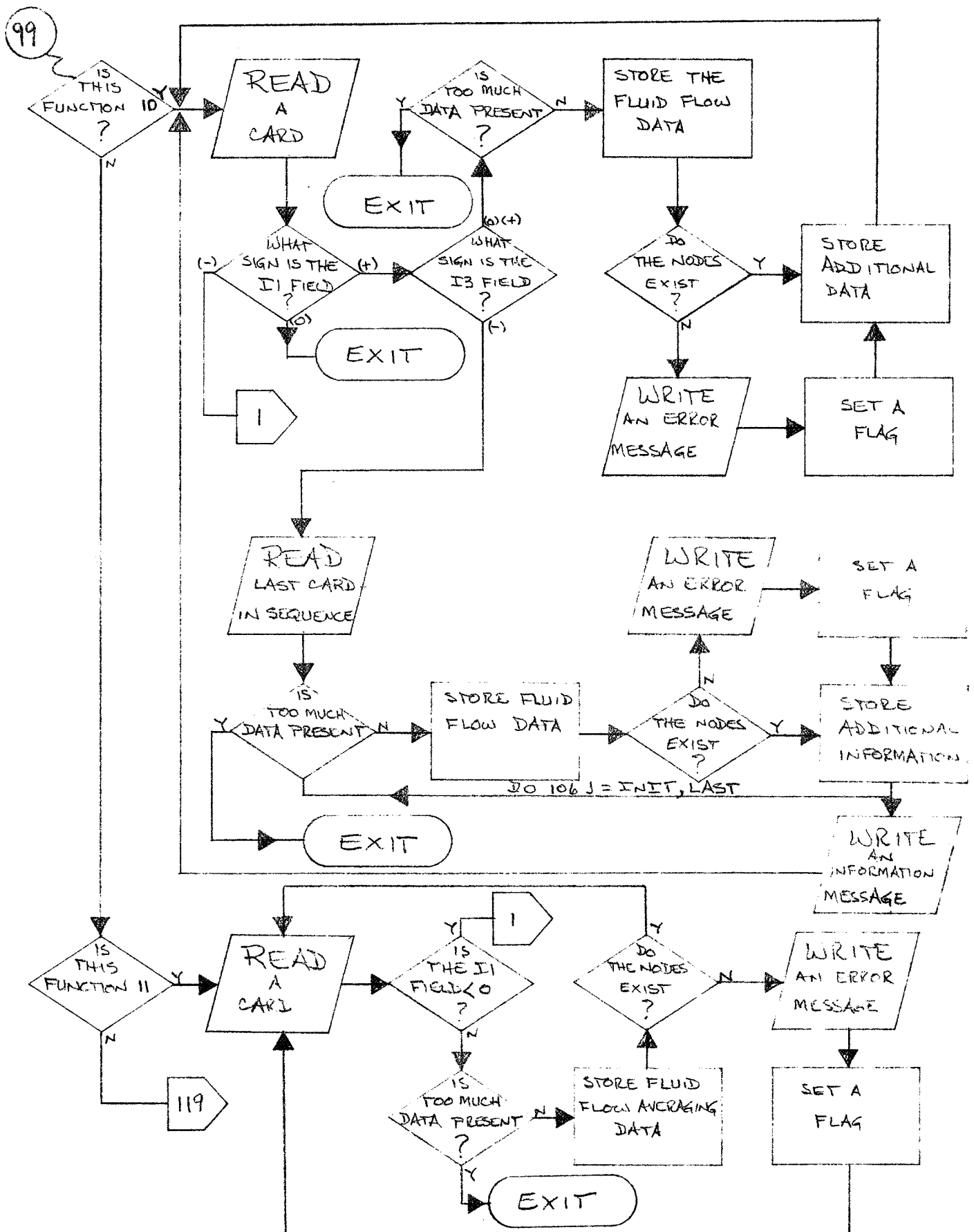




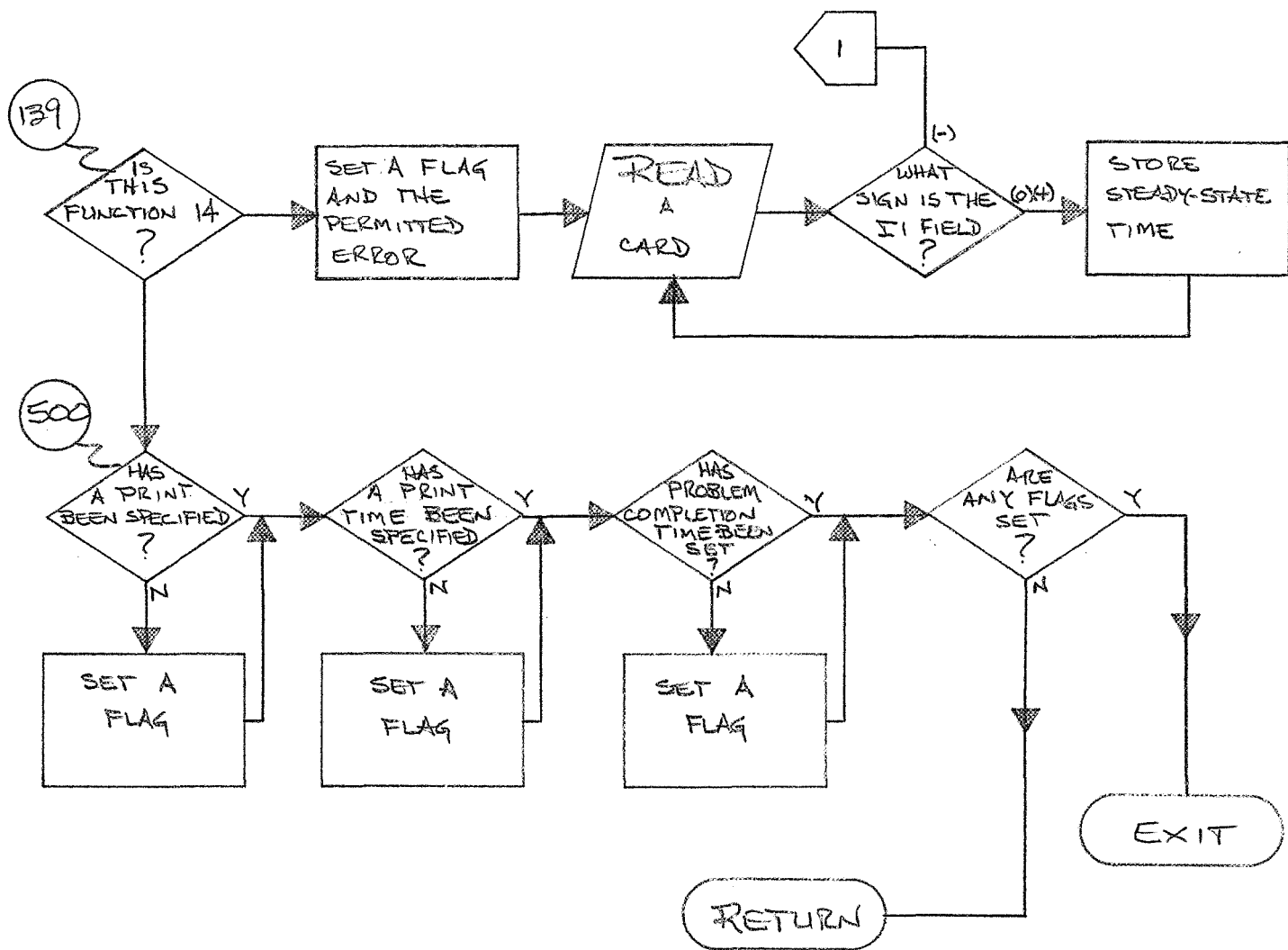


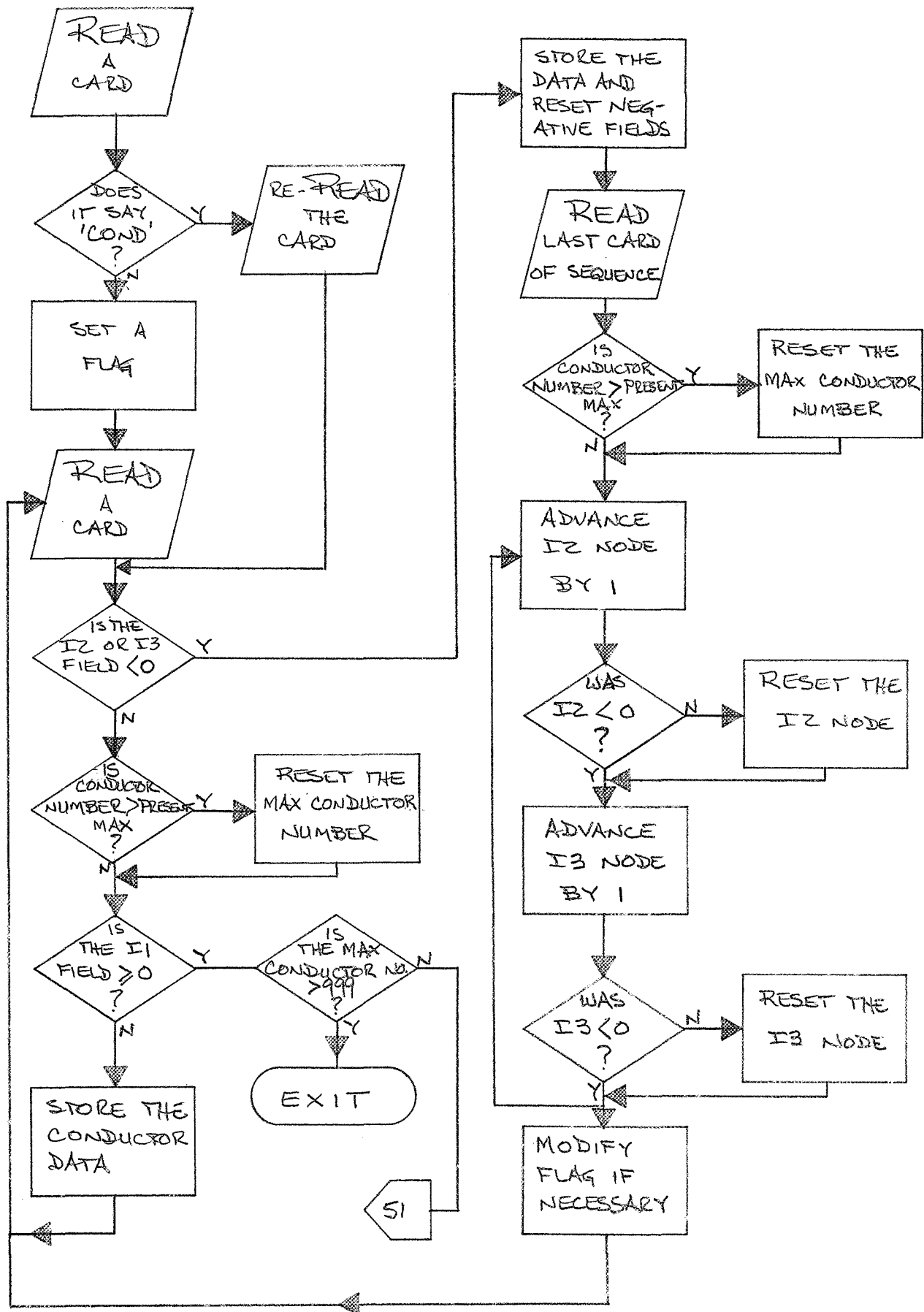


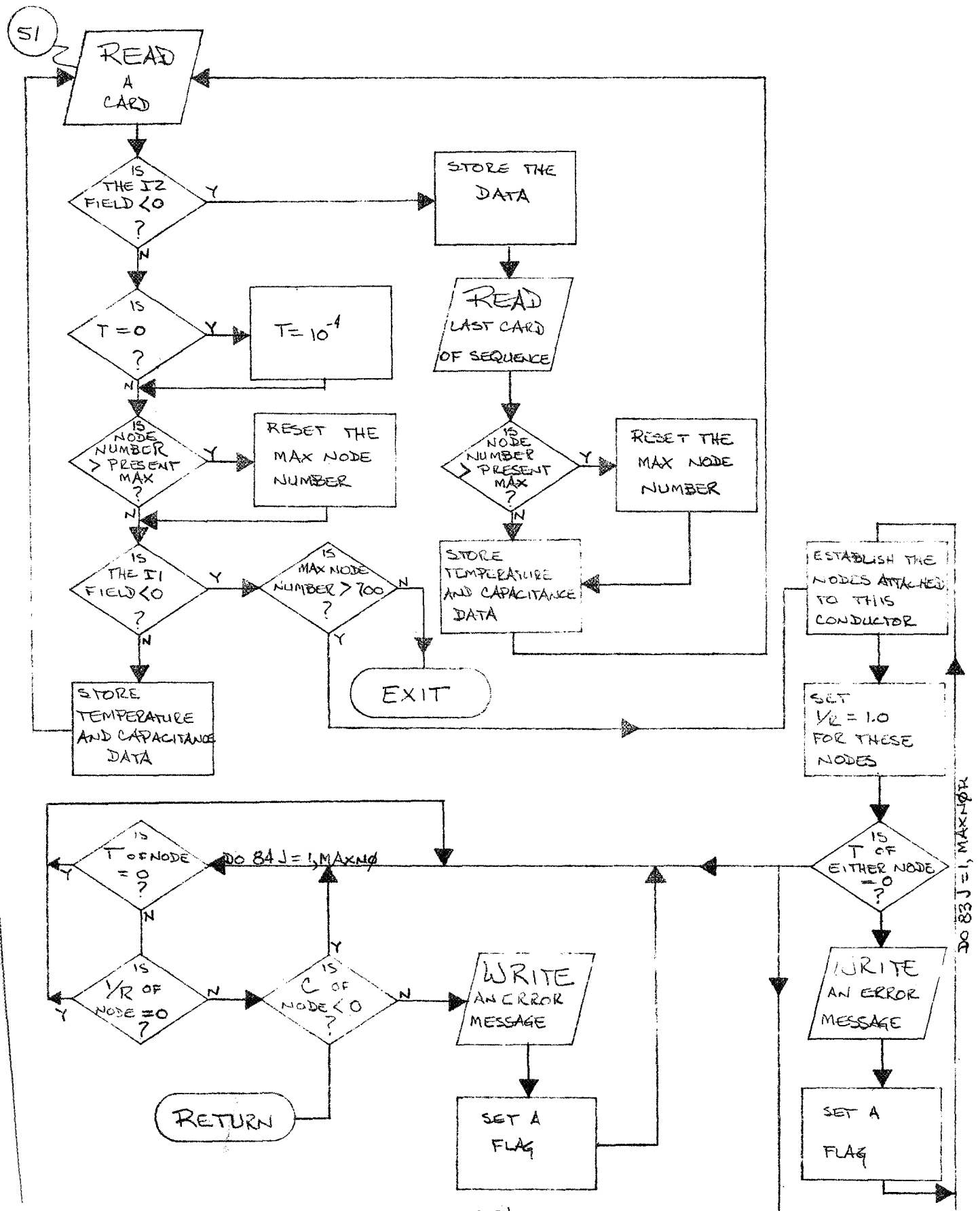


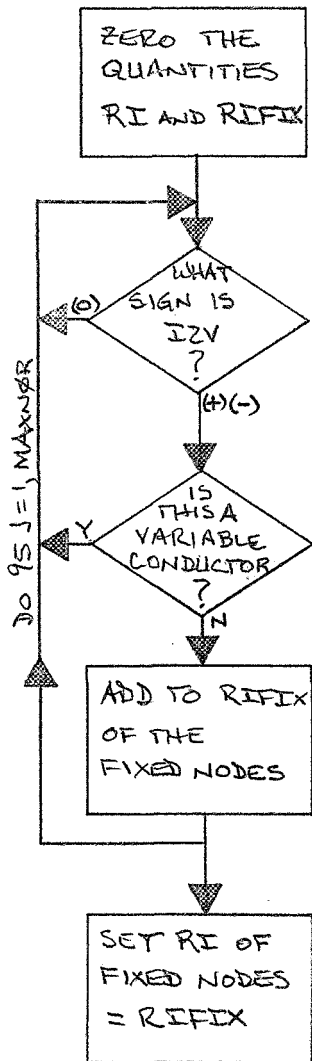
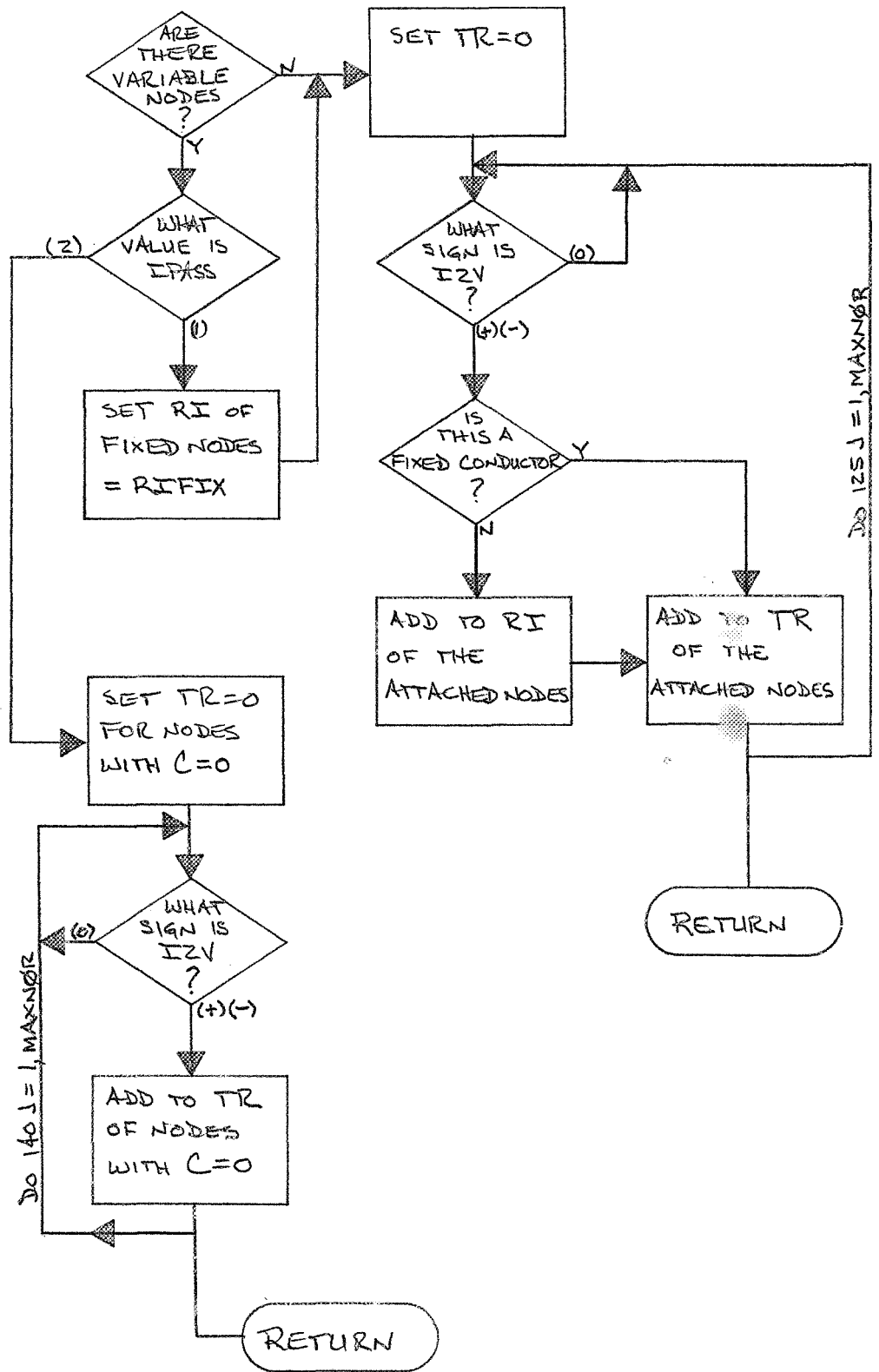


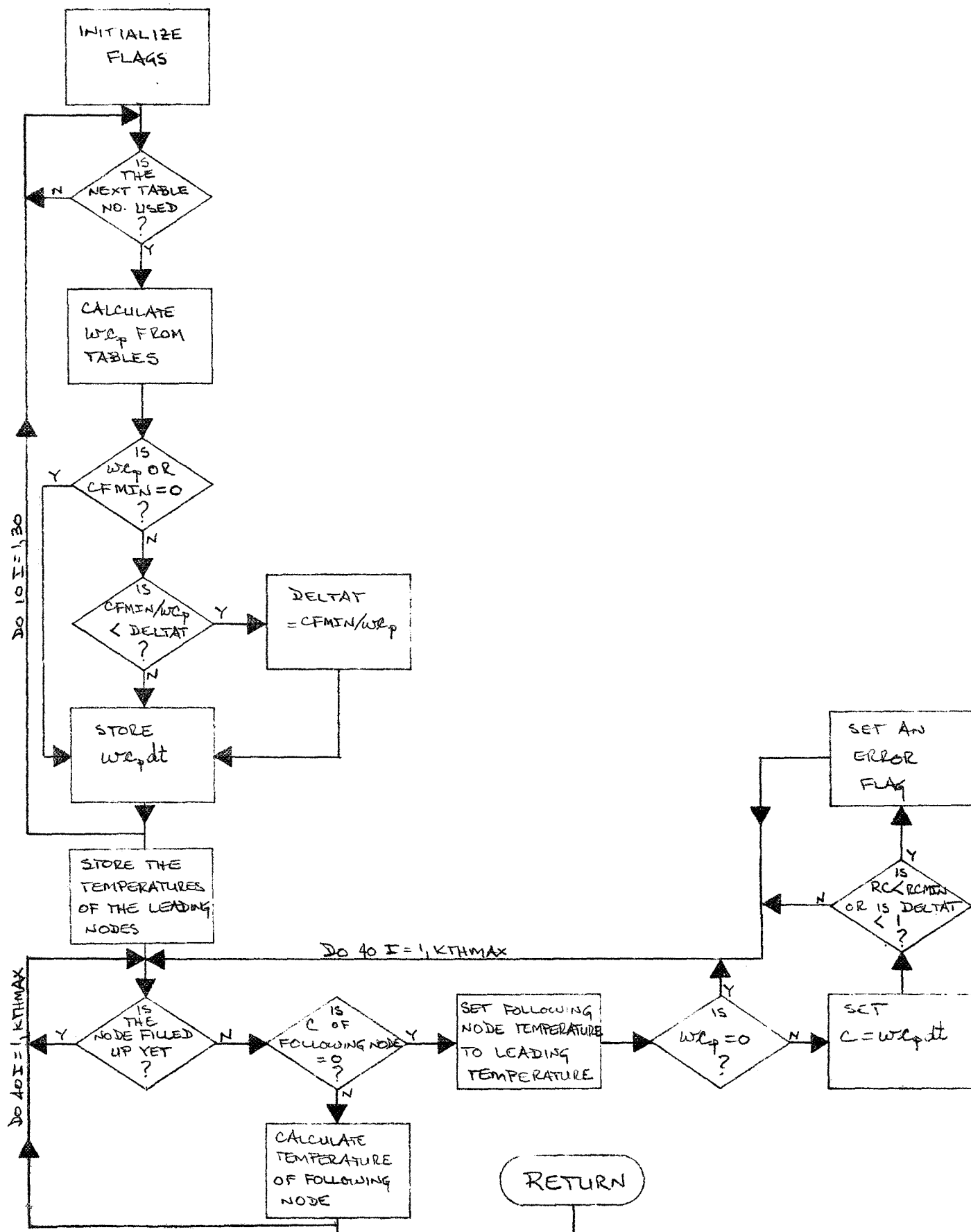


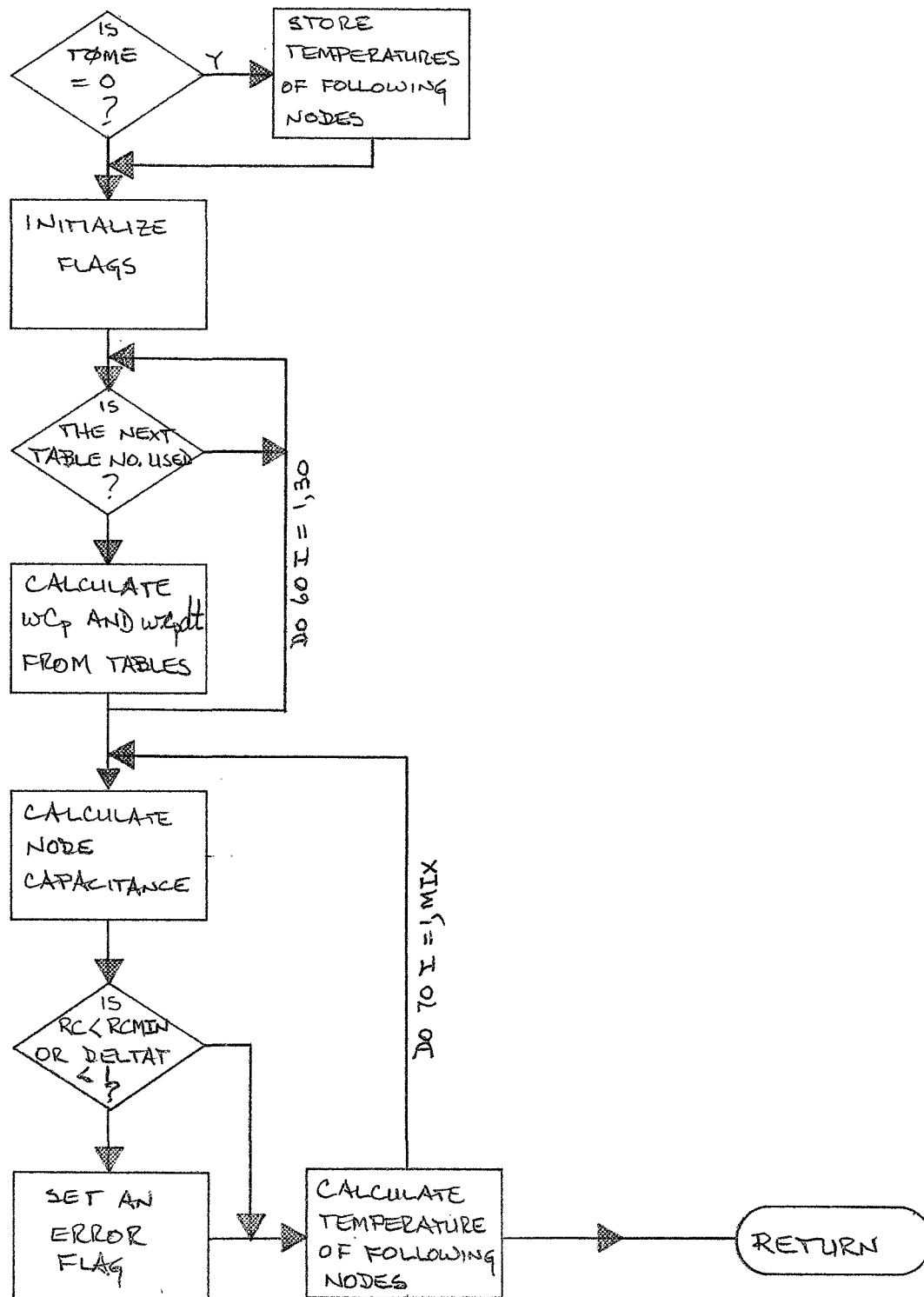


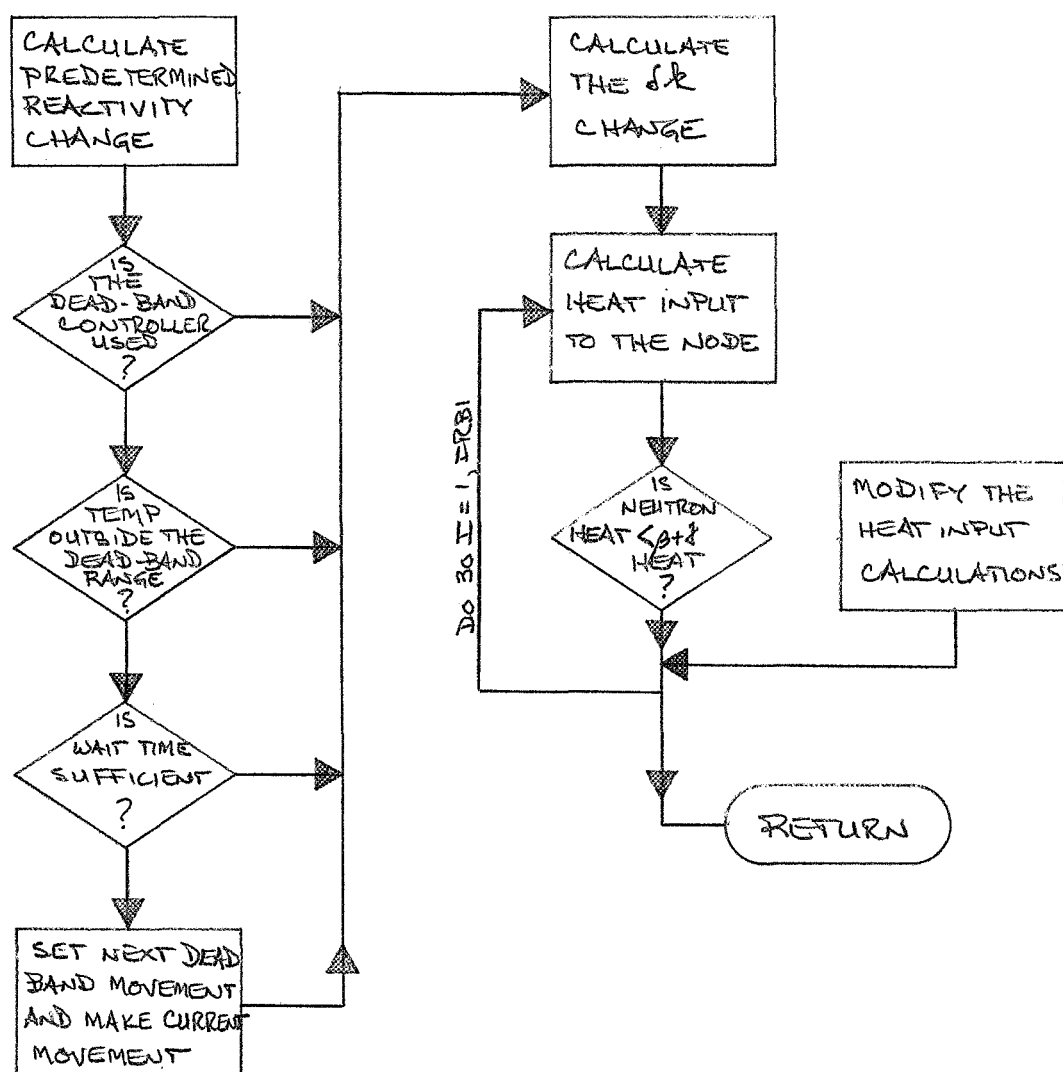


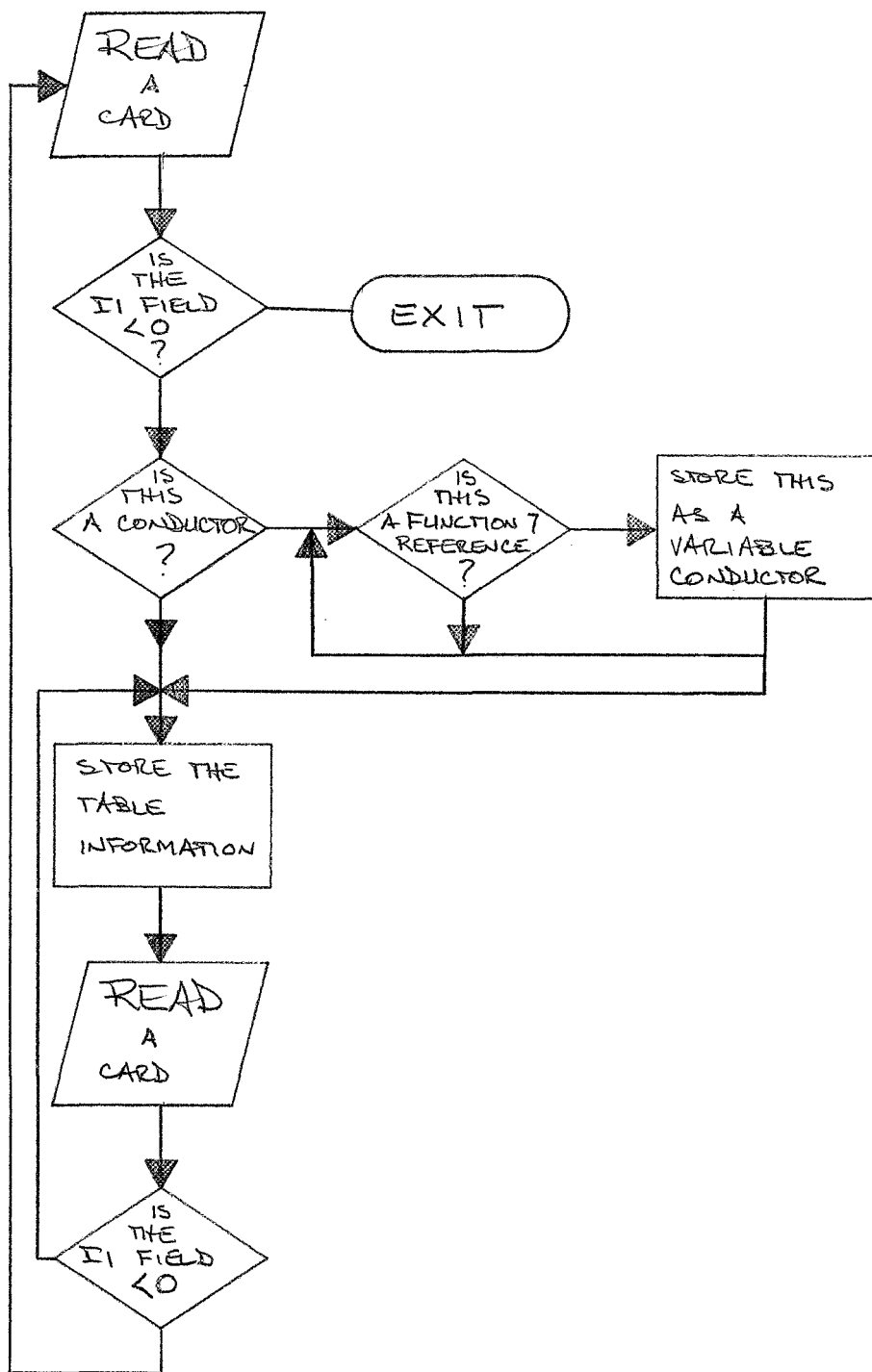


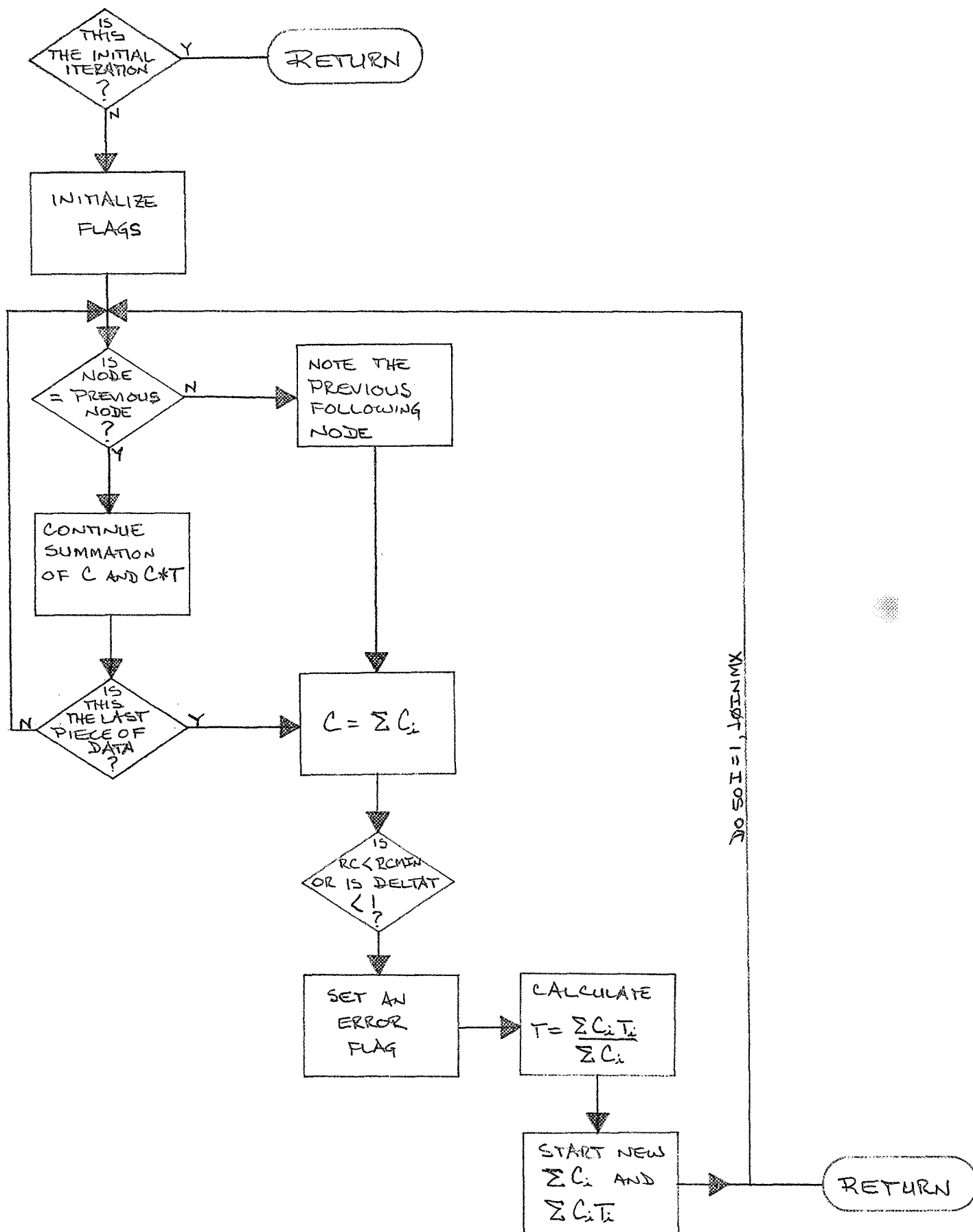
ENTRY RFIXENTRY TRA

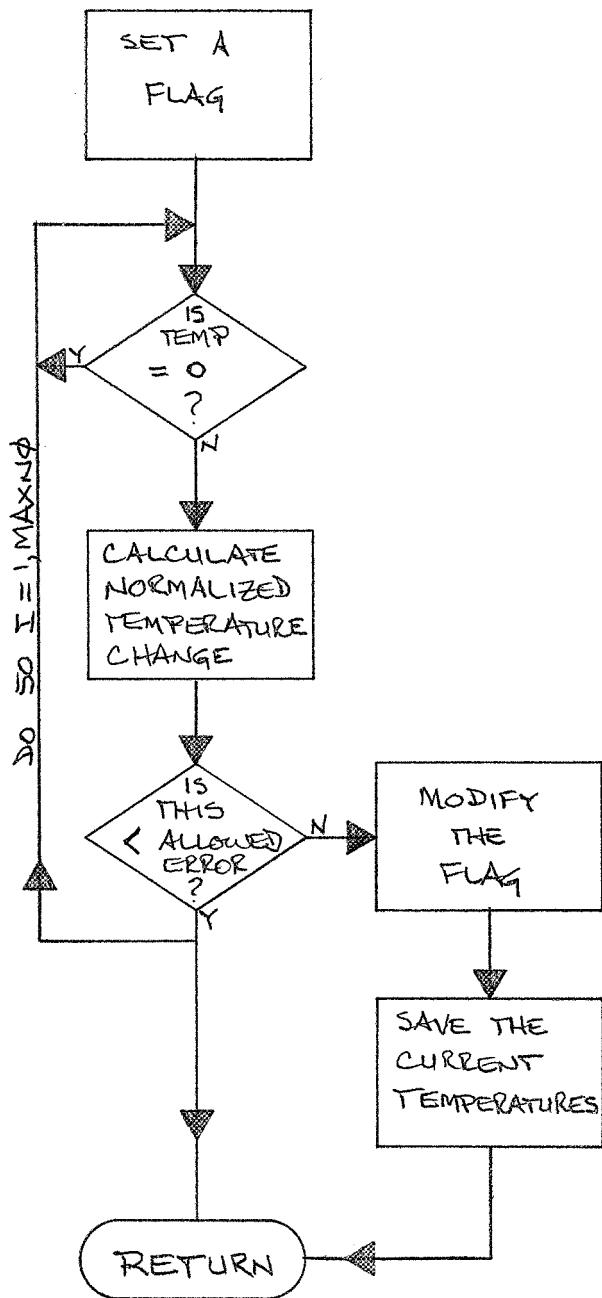


ENTRY FSTFL0

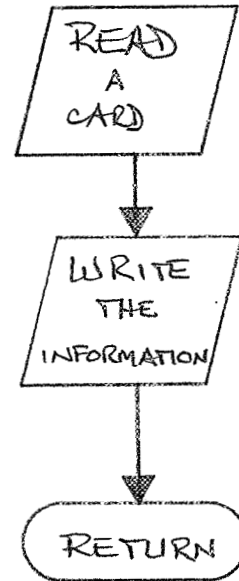
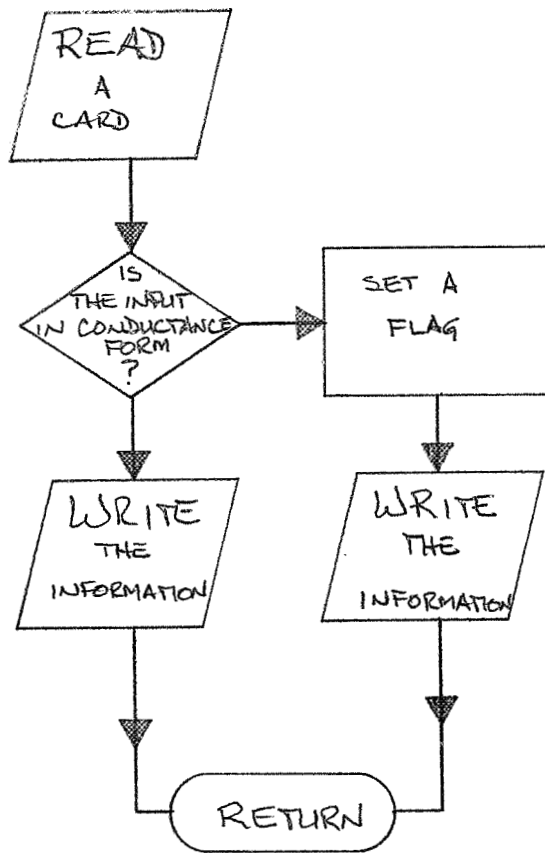


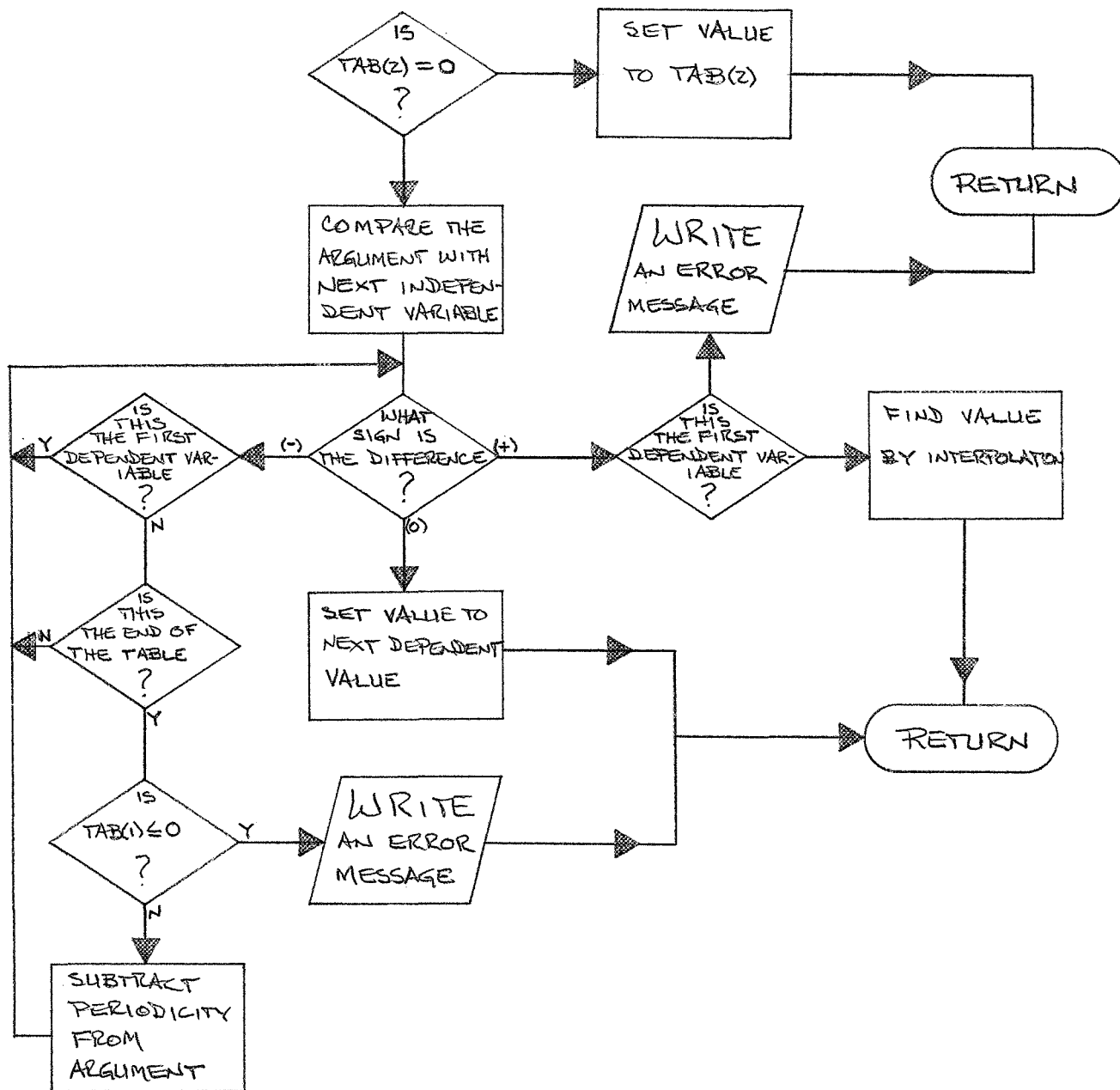






ENTRY READ2





A P P E N D I X B

```
      COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NODEIT,ID,XREF
ISN 0002      COMMON/FUN1/ICATH,KC(50)
ISN 0003      COMMON/FUN2/IRAD,KR(250),EKR(250)
ISN 0004      COMMON/FUN3/IPHK,IPH(50),PHT1(50),PHH1(50),PHT2(50),PHH2(50)
ISN 0005      COMMON/FUN4/ICONK,ICON(100),EXCON(100),ACON(100)
ISN 0006      COMMON/FUN5/IRB1,POWFAC(100),NDEQ(100),Q0,IDEK
ISN 0007      COMMON/FUN6/IAER1,IAER2,IAERK,AER(51),IAER(50),I2AER(50),ALLOC(50)
ISN 0008      COMMON/FUN7/IKF,IAF2(250),IAF3(250),IAF4(250),IAF5(250)
ISN 0009      COMMON/FUN8/IPK,IPR1(700),IPR2(700),PRUNT2(700)
ISN 0010      COMMON/FUN9/CO(20),PRTIME(5),TYM(5)
ISN 0011      COMMON/FUN10/KTHMAX,NEW(250),KLEAD(250),CEFLUID(250),KTABLE(30),
      *LTABLE(250),CFMIN(30),FULTIM(250)
ISN 0012      COMMON/FUN11/JOINMX,JTABLE(50),NUNODE(50),INNODE(50)
ISN 0013      COMMON/FUN12/MIX,KKTABL(30),LLEAD(200),LLTABL(200),LNEW(200),
      1CSOLID(200)
ISN 0014      COMMON/FUN14/F14,ERROR,NUTIME,ADTIME(20)
ISN 0015      COMMON/IARNO/INDEX
ISN 0016      COMMON/CRASH/SIGNAL
ISN 0017      COMMON/VARY/ NODVAR,IRVAR(999)
ISN 0018      REAL*8 CARD,TYP, T,C,Q,PRINT,TR,RI,RC,TPREV, COND,EX,IDIFF
ISN 0019      REAL*8 QBETA
ISN 0020      INTEGER*2 KC,KR,IPH,ICON,NODEQ,IAER,I2AER,IAF2,IAF3,IAF4,IPR1,
      1IPR2,NEW,KLEAD,KTABLE,LTABLE,JTABLE,NUNODE,INNODE,KKTABL,LLEAD,
      2LLTABL,LNEW,I2V,I3V,KIND,KDEP
ISN 0021      INTEGER*2 IRVAR,PRUNT2
ISN 0022      DIMENSION T(700),COND(999),C(700),TR(700),RI(700),RC(700),
      1I2V(999),I3V(999),Q(700),WCP(30),TPREV(50),QBETA(100),
      2TABLE(104,30),KIND(30),KDEP(30),QDEL(50),TITLE(18),PRUNT(700),
      3PRINT(3129,1)
ISN 0023      COMMON PRINT
ISN 0024      COMMON TR , RI , RC , I2V , I3V
ISN 0025      EQUIVALENCE (PRINT(1),COND(1)),(PRINT(1000),T(1))
      1(PRINT(1700),C(1)),(PRINT(2400),Q(1)),(PRINT(3100),WCP(1))
ISN 0026      CALL READRE
C
C ***** FORMAT STATEMENTS *****
C
ISN 0027      1001 FORMAT (A5)
ISN 0028      1002 FORMAT(5X,I1,18A4)
ISN 0029      2001 FORMAT(1H1,10X,18A4/1H0)
```

```

ISN 0030      2002 FORMAT(/,9H0  TIME =,F 9.2,5X,6HRCMIN(,I4,3H) =,IPE11.5,5X,
                *13HDELTA THETA =,IPE10.4)
                TAP16620
ISN 0031      2003 FORMAT(/,9H0  TIME =,F 9.2,5X,6HRCMIN(,I4,3H) =,IPE11.5,5X,
                *13HDELTA THETA =,IPE10.4,5X,9HDELTA K =,F11.6)
                TAP16660
ISN 0032      2004 FORMAT(/,5(5X,A2,I4,1H=,IPE12.5))
ISN 0033      2010 FORMAT(1H0,15X,26HSTEADY STATE VALUES FOLLOW)
ISN 0034      3001 FORMAT(I5,10X,2E15.5)
ISN 0035      9001 FORMAT (25H0COMPUTING INTERVAL SMALL13)
ISN 0036      9005 FORMAT(18H0RCMIN ZERO, NODE=,I3)
ISN 0037      9010 FORMAT(10X,50(,*)) /5X, 'THE JOB HAS BEEN CANCELLED BECAUSE OF A MAT
                1HEMATICAL INSTABILITY IN THE FUNCTION INDICATED',/10X, 'THE NEXT CAS
                2E WILL NOW BE PROCESSED IF ONE EXISTS')
ISN 0038      9999 FORMAT(1H1,20X,40(*)) /5X, 'THE RUN HAS BEEN TERMINATED BECAUSE OF
                1THE ERROR CONDITION NOTED')

```

C

C INITIALIZE PROGRAM VARIABLES

C

```

ISN 0039      DATA CARD/5HTITLE/
ISN 0040      LOGICAL SIGNAL
ISN 0041      1 READ (5,1001) TYP
ISN 0042      IF(TYP.NE.CARD) GO TO 1
ISN 0044      READ(99,1002) KPUNCH,TITLE
ISN 0045      WRITE (6,2001) TITLE
ISN 0046      DO 2J=1,30
ISN 0047      CO(J)=0.
ISN 0048      KIND(J)=0.
ISN 0049      KDER(J)=0.
ISN 0050      WCP(J)=0.
ISN 0051      KTABLE(J)=0.
ISN 0052      KKTABL(J)=0
ISN 0053      CFMIN(J)=1.E10
ISN 0054      DO 2I=1,104
ISN 0055      2 TABLE(I,J)=0.
ISN 0056      DO 3 I=1,5
ISN 0057      TYM(I)=0.
ISN 0058      3 PRTIME(I)=0.
ISN 0059      DO 5 I=1,50
ISN 0060      QDEL(I)=0.
ISN 0061      NUNODE(I)=0
                TAP11390
                TAP11410
                TAP1143
                TAP11450
                TAP11460
                TAP11490
                T171148
                TAP11500
                TAP11520
                TAP11530
                TAP11710
                TAP11750

```

ISN 0062	5 INNODE(I)=0	
ISN 0063	DO 7 I=1,700	
ISN 0064	T(I)=0.	TAP11910
ISN 0065	C(I)=0.	
ISN 0066	7 Q(I)=0.	
ISN 0067	DO 8 I=1,999	
ISN 0068	8 COND(J)=0.	
ISN 0069	SIGNAL=.TRUE.	TAP12120
	C CLEAR CORE LOCATIONS	TAP12180
ISN 0070	IERR=0	
ISN 0071	MAXNO=0	
ISN 0072	MAXNOR=0	
ISN 0073	MINND=0	
ISN 0074	KEY=0	TAP12350
ISN 0075	LI=1	TAP12360
ISN 0076	DELTAT=0.	TAP12370
ISN 0077	RCMIN=0.	
ISN 0078	PRTST=0.0	TAP12390
ISN 0079	DELK=0.	TAP12420
ISN 0080	DBTOT=0.	TAP12430
ISN 0081	DISAVE=0.	
ISN 0082	NSTEP=1	TAP12450
ISN 0083	ISTEP=0	
ISN 0084	LASTPR=1	
ISN 0085	IQME=0.	TAP12470
ISN 0086	NODVAR=0	
ISN 0087	MININT=0	
ISN 0088	TYM(1)=1.E10	TAP12490
ISN 0089	PRTIME(1)=1.E10	TAP12500
	C	TAP12510
ISN 0090	CALL TRCIN(MAXNDR,I2V,I3V,COND,MAXNO,I,C,RI,TR)	
	C	TAP12530
	C READ IN FUNCTION DATA	TAP12540
	C	TAP12550
ISN 0091	CALL FIN(T,I2V)	
ISN 0092	IF(IRAD.GT.0.OR.ICONK.GT.0) NODVAR=1	
ISN 0094	T2ME=CO(9)	
	C	
	C SET THE INITIAL PRINT TIME	

```

ISN 0095      10 ISTEP=ISTEP+1
ISN 0096      IF (T2ME.GE.PRTIME(ISTEP)) GO TO 10
ISN 0098      PRNT=CO(ISTEP+15)
ISN 0099      CO(1)=PRNT
C
C READ IN TABLES
C
ISN 0100      CALL TABIN(TABLE,KDEP,KIND)
ISN 0101      IF (.NOT.SIGNAL) GO TO 999
C
ISN 0103      CALL RFX
C
C SET UP BRANCHES FOR FUNCTION CALCULATIONS
C
ISN 0104      IF (IRAD)121,121,120
ISN 0105      120 ASSIGN 301 TO NU2
ISN 0106      GO TO 122
ISN 0107      121 ASSIGN 305 TO NU2
ISN 0108      122 IF (ICONK)124,124,123
ISN 0109      123 ASSIGN 306 TO NU3
ISN 0110      GO TO 125
ISN 0111      124 ASSIGN 310 TO NU3
ISN 0112      125 IF (IRB)1127,127,126
ISN 0113      126 ASSIGN 237 TO NU4
ISN 0114      GO TO 128
ISN 0115      127 ASSIGN 240 TO NU4
ISN 0116      128 IF (IAERK)132,132,129
ISN 0117      129 IAERK=IAERK-1
ISN 0118      IF (IAER)1130,130,131
ISN 0119      130 ASSIGN 311 TO NU5
ISN 0120      ASSIGN 320 TO NU6
ISN 0121      GO TO 133
ISN 0122      131 ASSIGN 315 TO NU5
ISN 0123      ASSIGN 316 TO NU6
ISN 0124      GO TO 133
ISN 0125      132 ASSIGN 320 TO NU5
ISN 0126      133 IF (IKF)135,135,134
ISN 0127      134 ASSIGN 291 TO NU7
ISN 0128      GO TO 136

```

TAP12610

TAP12620

TAP12630

TAP12640

TAP12660

TAP12670

TAP12680

TAP12690

TAP12750

TAP12790

TAP12900

TAP12930

TAP12970

ISN 0129	135 ASSIGN 300 TO NU7	
ISN 0130	136 IF (ICATH)138,138,137	
ISN 0131	137 ASSIGN 321 TO NU1	TAP13010
ISN 0132	GO TO 139	
ISN 0133	138 ASSIGN 325 TO NU1	
ISN 0134	139 IF (IPHK)142,142,140	
ISN 0135	140 ASSIGN 326 TO NU8	
ISN 0136	DO 141 I=1,IPHK	TAP13060
ISN 0137	INDEX=IPH(I)	
ISN 0138	141 TPREV(I)=T(INDEX)	
ISN 0139	GO TO 150	TAP13080
ISN 0140	142 ASSIGN 335 TO NU8	
ISN 0141	150 IF (KTHMAX)152,152,151	TAP13100
ISN 0142	151 ASSIGN 230 TO NU10	TAP13110
ISN 0143	GO TO 155	TAP13120
ISN 0144	152 ASSIGN 231 TO NU10	
ISN 0145	155 IF(JOINMX)157,157,156	TAP13140
ISN 0146	156 ASSIGN 281 TO NU11	TAP13150
ISN 0147	GO TO 160	TAP13160
ISN 0148	157 ASSIGN 282 TO NU11	
ISN 0149	160 IF(MIX) 162,162,161	TAP13180
ISN 0150	161 ASSIGN 266 TO NU12	
ISN 0151	GO TO 200	
ISN 0152	162 ASSIGN 270 TO NU12	
ISN 0153	200 ASSIGN 210 TO N2	
ISN 0154	WRITE (6,2001) TITLE	
ISN 0155	ASSIGN 330 TO NUT	
	C	TAP13360
	C GO TO BLOCK TO DO I/R AND T/R SUMMATION	TAP13370
	C	TAP13380
ISN 0156	205 CALL TRA(1)	
	C	TAP13410
	C CALCULATE DELTAT FOR THIS INTERVAL	
	C SET FLAGS TO PERFORM ONLY NECESSARY CALCULATIONS	
	C	TAP13430
ISN 0157	GO TO N2,(210,216)	
ISN 0158	210 DO 214 J=1,MAXNO	
ISN 0159	IF(T(J)) 211,214,211	
ISN 0160	211 IF(C(J)) 214,212,214	

TAP13480

212 T(J)=TR(J)/RI(J)

214 CONTINUE

216 ASSIGN 216 TO N2

GO TO 400

216 RCMIN=1.E10

DO 220 J=1,MAXND

IF(C(J)) 220,220,218

218 RC(J)=C(J)/RI(J)

IF(RC(J).GT.RCMIN) GO TO 220

RCMIN=RC(J)

MINND=J

220 CONTINUE

IF(RCMIN.LE.0.) GO TO 905

C

C IF DELTAT IS PRESELECTED, ESTABLISH WHICH DELTAT IS CURRENTLY USED

C IF DELTAT IS TO BE AUTOMATICALLY SELECTED, DELTAT=RCMIN

C

IF(CO(2)) 225,225,223

223 IF(I2ME.GE.TYM(NSSTEP)) NSTEP=NSTEP+1

DELTAT=CO(NSSTEP+10)

IF(DELTAT.GT.RCMIN) GO TO 225

GO TO 228

225 DELTAT=RCMIN

IF(CO(3)) 228,228,227

227 DELTAT=CO(3)*DELTAT

228 IF(CO(10)-DELTAT) 229,229,900

C

C PERFORM FLUID FLOW CALCULATIONS (FUNCTION 10)

C

229 GO TO NU10,(230,231)

230 CALL FLOW(TOME,WCP,I2ME,TABLE,DELTAT,T,C,RI,RCMIN,IERR)

IF(IERR.EQ.1) GO TO 910

IF(.NOT.SIGNAL) GO TO 999

C

C CHECK TO DETERMINE IF A PRINT IS REQUIRED ON THIS ITERATION

C

231 IF(I2ME.LE.PRTIME(ISTEP)) GO TO 232

ISTEP=ISTEP+1

LASTPR=1

ISN 0193

ISN 0195

ISN 0196

ISN 0197

ISN 0198

ISN 0199

ISN 0200

```

ISN 0197      PRISI=0,
ISN 0198      CO(1)=CO(ISTEP+15)
ISN 0199      PRNT=PRIME(ISTEP-1)+CO(1)
ISN 0200      232 PRISI=-T2ME-DELTA+PRNT
ISN 0201      IF(PRISI) 233,234,235
ISN 0202      233 DELTA=DELTA+PRISI
ISN 0203      234 LASTPR=LASTPR+1
ISN 0204      ADD=LASTPR
ISN 0205      PRNT=ADD*CO(1)
ISN 0206      IF(ISTEP.GT.1) PRNT=PRNT+PRIME(ISTEP-1)
ISN 0208      PRISI=0.
ISN 0209      235 TFLAG=1.0
ISN 0210      GO TO NU4, (237,240)
                                     C
                                     C CALL IN THE HEATING FUNCTION
                                     C
ISN 0211      237 CALL HEATIN(T2ME,TABLE,OBETA,I,Q,DBICT)
ISN 0212      IF(.NOT.SIGNAL) GO TO 999
                                     C
                                     C NOW PERFORM THE PRIMARY TEMPERATURE CALCULATIONS
                                     C
ISN 0214      240 DO 245 J=1,MAXNO
ISN 0215      IF(T(J)) 241,245,241
ISN 0216      241 IF(C(J)) 245,242,244
ISN 0217      242 TFLAG=0.0
ISN 0218      GO TO 245
ISN 0219      244 T(J)= DELTA/C(J)*(Q(J)+TR(J)-T(J)*RI(J))+T(J)
ISN 0220      IF(T(J).EQ.0.) T(J)=0.001
ISN 0222      245 CONTINUE
ISN 0223      IF(TFLAG) 256,256,265
                                     C
                                     C GO TO INSTR BLOCK TO DO I/R AND T/R SUMMATION FOR SECOND PASS THIS
                                     C INTERVAL TO RELAX ZERO CAPACITANCE NODES.
                                     C
ISN 0224      256 CALL TRA(2)
ISN 0225      DO 260 J=1,MAXNO
ISN 0226      IF(T(J)) 257,260,257
ISN 0227      257 IF(C(J)) 260,258,260
ISN 0228      258 T(J)=(Q(J)+TR(J))/RI(J)
TAP14120
TAP14140
TAP14670
TAP14680
TAP14690
TAP14700

```


ISN 0229	IF (T(J)) 260,259,260	
ISN 0230	259 T(J)=.0001	
ISN 0231	260 CONTINUE	TAP14800 TAP14810 TAP14820
	C	
	C	
	C	
ISN 0232	265 GO TO NULL,(266,270)	
ISN 0233	266 CALL FSTFLO(TCME,WCP,T2ME,TABLE,DELTAT,T,C,Q,TR,RI,RCMIN,IERR)	
ISN 0234	IF(IERR.EQ.1) GO TO 910	
ISN 0236	IF(.NOT.SIGNAL) GO TO 999	
	C	
	C	
	C	
	C	
ISN 0238	270 T2ME=T2ME+DELTAT	
	C	
	C	
	C	
	C	
	C	
ISN 0239	IF(F14.NE. 1.) GO TO 280	
ISN 0241	CALL STEADY(KEY,I,MAXNO,DELTAT)	TAP15080
ISN 0242	IF(KEY.EQ.0) GO TO 280	
ISN 0244	WRITE(6,2010)	
ISN 0245	IF(NUTIME.EQ.0) GO TO 273	
ISN 0247	271 IF(T2ME.LT.ADTIME(LL)) GO TO 272	
ISN 0249	LL=LL+1	TAP15130
ISN 0250	GO TO 271	
ISN 0251	272 T2ME=ADTIME(LL)	
ISN 0252	LL=LL+1	
ISN 0253	IF(LL.GT.NUTIME) NUTIME=0	TAP15170
ISN 0255	GO TO 400	
ISN 0256	273 T2ME=CO(4)+1.	
ISN 0257	GO TO 400	
	C	
	C	
	C	
	C	
ISN 0258	280 GO TO NULL,(281,282)	TAP15220 TAP15230 TAP15240
ISN 0259	281 CALL TAVG(TCME,C,T,WCP,DELTAT,RI,RCMIN,IERR)	
ISN 0260	IF(IERR.EQ.1) GO TO 910	
ISN 0262	282 TCME=1.	
ISN 0263	IF(PRTST.EQ. 0.) GO TO 400	

TAP15480
TAP15490
TAP15500

EVALUATION OF ALL VARIABLE GS,TS,RS AND CS AT EACH TIME INCREM.

ISN 0265 290 OTSAVE=DELTAT
ISN 0266 GO TO NU7, (291,300)

PERFORM ARBITRARY FUNCTIONS INTERPOLATIONS

TAP15520
TAP15530
TAP15540

ISN 0267 291 DO 297 I=1,IKF

ISN 0268 J=IAF2(I)
ISN 0269 L=IAF3(I)

ISN 0270 INDEX=IAF4(I)
ISN 0271 M=KIND(INDEX)

ISN 0272 NYS=KDEP(INDEX)
ISN 0273 MA = (NYS-1)*999 + J

ISN 0274 IF(NYS.GT.2) MA=MA-(NYS-2)*299
ISN 0276 MB = (M-1) * 999 + L

ISN 0277 IF(M.GT.2) MB=MB-(M-2)*299
ISN 0278 MC=(INDEX-1)*104+1

ISN 0280 IF (M) 294,294,293
ISN 0281 293 PRINT(MA,1)=ENTERP(PRINT(MB,1),TABLE(MC,1))

ISN 0282 IF(M.EQ.1) PRINT(MA,1)=1.0/PRINT(MA,1)
ISN 0284 GO TO 295

ISN 0285 294 PRINT(MA,1)=ENTERP(T2ME,TABLE(MC,1))
ISN 0286 295 IF(.NOT.SIGNAL) GO TO 999

ISN 0288 IF(FAF5(I)) 296,297,296
ISN 0289 296 PRINT(MA,1)=FAF5(I)*PRINT(MA,1)

ISN 0290 297 CONTINUE
ISN 0291 300 GO TO NU2,(301,305)

COMPUTE RADIATION RESISTORS

TAP15740
TAP15750
TAP15760

ISN 0292 301 DO 302 I=1,IRAD

ISN 0293 INDEX=KR(I)
ISN 0294 K2=I2V(INDEX)

ISN 0295 K3=I3V(INDEX)
ISN 0296 COND(INDEX)=(EKR(I))*((T(K2)+459.69)**2+(T(K3)+459.69)**2)*((T(K2)+

8459.69)+(T(K3)+459.69)))

ISN 0297 302 CONTINUE

ISN 0298 305 GO TO NU3,(306,310)

C

C COMPUTE CONVECTION RESISTORS

TAP15860
TAP15870
TAP15880

C

ISN 0299 306 G=ENTERP(T2ME, TABLE(1,2))
ISN 0300 IF(.NOT.SIGNAL) GO TO 999
ISN 0302 DO 307 I=1, ICONK

ISN 0303 INDEX=ICON(I)
ISN 0304 K2=I2V(INDEX)
ISN 0305 K3=I3V(INDEX)
ISN 0306 EX=EXCON(I)

TAP15920
TAP15930
TAP15940
TAP15950
TAP15960

ISN 0307 TDIFF=DABS(T(K2)-T(K3))
ISN 0308 COND(INDEX)=(G*ENTERP((T(K2)+T(K3))/2.0, TABLE(1,1)))*ACON(I)*
RSGNL(TDIFF**EX)

ISN 0309 IF(.NOT.SIGNAL) GO TO 999

ISN 0311 307 CONTINUE

ISN 0312 310 GO TO NU5,(311,315,320)

C

C COMPUTE AERODYNAMIC HEATING USING THE ECKERT METHOD

TAP16030
TAP16040
TAP16050

B
F

ISN 0313 311 FLAG=0.0

ISN 0314 DO 314 I=1, IAERK

ISN 0315 INDEX=IAER(I)

ISN 0316 O(INDEX)=

1 QECK(T2ME, FLAG, T(INDEX),

2 TABLE(1,3), TABLE(1,4), TABLE(1,11),

TABLE(1,5), TABLE(1,8), IAER2, IAER(I), ALLOC(I),

TABLE(1,9), TABLE(1,10), TABLE(1,6), CO(5), CO(6), CO(7), CO(8),

3TABLE(1,7), AER(I))

ISN 0317 IF(.NOT.SIGNAL) GO TO 999

ISN 0318 314 CONTINUE

ISN 0320 315 GO TO NU6,(316,320)

C

C COMPUTE AERO HEATING USING RAMO-WCCLDRIEGE METHOD

TAP16180
TAP16190
TAP16200

ISN 0321 316 FLAG=0.0

ISN 0322 DO 318 J=1, IAERK

ISN 0323 INDEX=IAER(J)

ISN 0324 O(INDEX)=

TAP16230

```

1   ORW(T2ME,FLAG,IAER2,T(INDEX),          TAP16250
2   TABLE(1,3),TABLE(1,4),TABLE(1,11),    TAP16260
   JCO(5),CO(6),CO(7),CO(8),AER(I),TABLE(1,5), TAP16270
2   I2AER(I),ALLOC(I),                    TAP16280
3   TABLE(1,6),TABLE(1,12))              TAP16290
   IF(.NOT.SIGNAL) GO TO 999
318  CONTINUE
   320 GO TO NU1,(321,325)
C
C PERFORM CATHODE FOLLOWER COMPUTATION
C
321 DO 322 I=2,ICATH,2
   K1=KC(I-1)
   K2=KC(I)
322 T(K1)=T(K2)
325 GO TO NU8,(326,335)
C
C COMPUTE LATENT HEAT EFFECTS
C
326 DO 327 I=1,IPHK
   INDEX=IPH(I)
327 CALL LAT(PHT1(I),PHH1(I),PHT2(I),PHH2(I),QDEL(I),T(INDEX),
   TPREV(I),
   1   2Q(INDEX),RI(INDEX),TR(INDEX),DELTAT,T2ME,INDEX)
   IF(.NOT.SIGNAL) GO TO 999
   GO TO 335
C
C TEST FOR END OF JOB
C
330 CONTINUE
   ASSIGN 290 TO NUT
   IF(TOME.EQ. 0.) GO TO 290
   IF(MININT-1) 335,1,335
335 IF(T2ME-CO(4))205,336,336
336 CONTINUE
   IF(KPUNCH.NE. 1) GO TO 1
   DO 340 I=1,MAXNO
   IF(T(I).EQ. 0.) GO TO 340
   WRITE(7,3001) I,T(I),C(I)

```



```
COMPILER OPTIONS - NAME= TAP7,DPT=0.2,LINECNT=41, SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NODEIT, ID,XREF
ISN 0002 SUBROUTINE TRCIN(MAXNOR,I2V,I3V,CCND,MAXND,T,C,RI,TR)
C THIS SUBROUTINE READS IN ALL RESISTOR, TEMPERATURE, CAPACITOR DATA TRCII040
C TRCI1050
ISN 0003 REAL*8 CCND,T,C,RI,TR
ISN 0004 REAL*8 RIFIX(700)
ISN 0005 INTEGER*2 I2V,I3V
ISN 0006 INTEGER*2 IRVAR
ISN 0007 COMMON/CRASH/SIGNAL
ISN 0008 COMMON/VARY/ NODVAR,IRVAR(999)
ISN 0009 COMMON/FLAG/ KELAG
ISN 0010 DIMENSION I(3),I2V(999),I3V(999),CCND(999),T(700),C(700),RI(700),
1TR(700)
ISN 0011 EQUIVALENCE(I(1),I1),(I(2),I2),(I(3),I3)
ISN 0012 DATA CSET/,COND/,CCHECK/,C,/
C
C READ RESISTOR DATA
C
ISN 0013 KELAG=-1
ISN 0014 DO 1 J1=1,999
ISN 0015 1 I2V(J1)=0
C
ISN 0016 READ(5,1002) CON
ISN 0017 IF(CON.NE.CSET) GO TO 2
ISN 0019 KELAG=1
ISN 0020 GO TO 10
ISN 0021 2 READ(99,1003) CON,I,RIN
ISN 0022 WRITE(6,1003) CON,I,RIN
ISN 0023 IF(CON.EQ.CCHECK) KELAG=0
ISN 0025 GO TO 11
ISN 0026 10 CALL READ(I,RIN)
ISN 0027 11 IF(I2.LT.0.OR.I3.LT.0) GO TO 20
ISN 0029 MAXNOR=MAX0(MAXNOR,I1)
ISN 0030 IF(I1) 50,50,15
ISN 0031 15 M=I1
ISN 0032 I2V(M)=I2
ISN 0033 I3V(M)=I3
ISN 0034 COND(M)=1.0/RIN
TRCII190
TRCII200
TRCII210
TRCII220
TRCII230
```

ISN 0035	IF(KFLAG.EQ.0) COND(M)=RIN	
ISN 0037	IF(KFLAG.EQ.0) KFLAG=-1	TRC11250
ISN 0039	GO TO 10	TRC11260
ISN 0040	20 K1=I1	
ISN 0041	K2SAVE=I2	
ISN 0042	K3SAVE=I3	
ISN 0043	IF(I2.LT.0) K2=-I2	
ISN 0045	IF(I3.LT.0) K3=-I3	
ISN 0047	CALL READ(I,RIN)	
ISN 0048	MAXNOR=MAX0(MAXNOR,I1)	TRC11300
ISN 0049	K1MAX=I1	TRC11310
ISN 0050	K2I=K2-K1	TRC11320
ISN 0051	K3I=K3-K1	TRC11330
ISN 0052	DO 25 J=K1,K1MAX	TRC11340
ISN 0053	I2V(J)=K2I+J	TRC11350
ISN 0054	IF(K2SAVE.GT.0) I2V(J)=K2SAVE	
ISN 0056	I3V(J)=K3I+J	TRC11360
ISN 0057	IF(K3SAVE.GT.0) I3V(J)=K3SAVE	
ISN 0059	COND(J)=1.0/RIN	
ISN 0060	IF(KFLAG.EQ.0) COND(J)=RIN	TRC11380
ISN 0062	IF(J.EQ.K1.OR.J.EQ.K1MAX) GO TO 25	TRC11390
ISN 0064	WRITE(6,1000) J,I2V(J),I3V(J),RIN	TRC11400
ISN 0065	25 CONTINUE	
ISN 0066	IF(KFLAG.EQ.0) KFLAG=-1	TRC11410
ISN 0068	GO TO 10	
ISN 0069	50 IF(MAXNOR.LE.999) GO TO 51	
ISN 0071	WRITE(6,2001)	
ISN 0072	SIGNAL=.FALSE.	TRC11420
		TRC11430
		TRC11440
ISN 0073	51 CALL READ2(I,TIN,CIN)	TRC11460
		TRC11470
		TRC11480
ISN 0074	IF(I2.LT.0) GO TO 70	TRC11490
ISN 0076	IF(TIN.EQ.0) TIN=.0001	TRC11500
ISN 0078	MAXNO=MAX0(MAXNO,I1)	TRC11510
ISN 0079	IF(I1) 80,80,60	

ISN 0080	60 M=11	TRC11530
ISN 0081	T(M)=TIN	TRC11540
ISN 0082	C(M)=CIN	TRC11550
ISN 0083	GO TO 51	
ISN 0084	70 K1=11	TRC11570
ISN 0085	CALL READ2(I,TIN,CIN)	TRC11580
ISN 0086	MAXNO=MAX0(MAXNO,11)	TRC11590
ISN 0087	K1MAX=11	TRC11600
ISN 0088	IF(TIN.EQ. 0) TIN=.0001	TRC11610
ISN 0090	DO 75 J=K1,K1MAX	TRC11620
ISN 0091	T(J)=TIN	TRC11630
ISN 0092	C(J)=CIN	TRC11640
ISN 0093	IF(J.EQ.K1.OR.J.EQ.K1MAX) GO TO 75	TRC11650
ISN 0095	WRITE (6,1001) J,TIN,CIN	TRC11660
ISN 0096	75 CONTINUE	TRC11670
ISN 0097	GO TO 51	
ISN 0098	80 IF(MAXNO.LE.700) GO TO 81	
ISN 0100	WRITE(6,2006)	
ISN 0101	SIGNAL=.FALSE.	
ISN 0102	RETURN	

B-16

C
C CHECK TO SEE IF ANY CONDUCTORS ARE ATTACHED TO NON-EXISTENT NODES OR
C IF ANY SPECIFIED NODES ARE NOT ATTACHED TO CONDUCTORS

ISN 0103	81 DO 83 J=1,MAXNDR	
ISN 0104	IF(I2V(J).EQ.0) GO TO 83	
ISN 0106	NODE1=I2V(J)	
ISN 0107	NODE2=I3V(J)	
ISN 0108	RI(NODE1)=1.	
ISN 0109	RI(NODE2)=1.	
ISN 0110	IF(I(NODE1).NE.0.) GO TO 82	
ISN 0112	WRITE(6,9001) J,NODE1	
ISN 0113	SIGNAL=.FALSE.	
ISN 0114	82 IF(T(NODE2).NE.0.) GO TO 83	
ISN 0116	WRITE(6,9001) J,NODE2	
ISN 0117	SIGNAL=.FALSE.	
ISN 0118	83 CONTINUE	
ISN 0119	DO 84 J=1,MAXNO	
ISN 0120	IF(I(J).EQ.0.) GO TO 84	


```

ISN 0122      IF(RI(J).NE.0.) GO TO 84
ISN 0124      IF(C(J).LT.0.) GO TO 84
ISN 0126      WRITE(6,9002) J
ISN 0127      SIGNAL=.FALSE.
ISN 0128      84 CONTINUE
ISN 0129      RETURN

```

C

C SECTION RFIX IS TO SET THE 1/R SUMS FOR THE NODES WITH FIXED RESISTORS

C

```

ISN 0130      ENTRY RFIX
ISN 0131      DO 85 J=1,MAXNO
ISN 0132      RI(J)=0.
ISN 0133      85 RFIX(J)=0.
ISN 0134      DO 95 J=1,MAXNOR
ISN 0135      IF(I2V(J)) 88,95,88
ISN 0136      88 I=IRVAR(J)
ISN 0137      IF(I) 95,91,95
ISN 0138      91 NODE1=I2V(J)
ISN 0139      NODE2=I3V(J)
ISN 0140      RFIX(NODE1)=RFIX(NODE1)+CCND(J)
ISN 0141      RFIX(NODE2)=RFIX(NODE2)+CCND(J)
ISN 0142      95 CONTINUE
ISN 0143      DO 100 J=1,MAXNO
ISN 0144      100 RI(J)=RFIX(J)
ISN 0145      RETURN

```

B-17

TRCI1700

C

C SECTION TRA IS TO ESTABLISH THE 1/R AND T/R SUMS FOR ALL NODES

C

TRCI1720

```

ISN 0146      ENTRY TRA(IPASS)
ISN 0147      IF(NODVAR) 103,111,103
ISN 0148      103 GO TO (105,130),IPASS
ISN 0149      105 DO 110 J=1,MAXNOR
ISN 0150      IF(IRVAR(J)) 110,110,106
ISN 0151      106 K=IRVAR(J)
ISN 0152      NODE1=I2V(K)
ISN 0153      NODE2=I3V(K)
ISN 0154      RI(NODE1)=RFIX(NODE1)
ISN 0155      RI(NODE2)=RFIX(NODE2)
ISN 0156      110 CONTINUE

```

```

ISN 0157      111 DO 120 J=1,MAXNO
ISN 0158      TR(J)=0.0
ISN 0159      120 CONTINUE
ISN 0160      DO 125 J=1,MAXNOR
ISN 0161      IF(I2V(J)) 122,125,122
ISN 0162      122 NODE1=I2V(J)
ISN 0163      NODE2=I3V(J)
ISN 0164      IF(IRVAR(J)) 123,124,123
ISN 0165      123 RI(NODE1)=RI(NODE1)+COND(J)
ISN 0166      RI(NODE2)=RI(NODE2)+COND(J)
ISN 0167      124 TR(NODE1)=TR(NODE1)+COND(J)*T(NODE2)
ISN 0168      TR(NODE2)=TR(NODE2)+COND(J)*T(NODE1)
ISN 0169      125 CONTINUE
ISN 0170      RETURN
ISN 0171      130 DO 131 J=1,MAXNO
ISN 0172      IF(C(J).EQ.0.) TR(J)=0.
ISN 0174      131 CONTINUE
ISN 0175      DO 140 J=1,MAXNOR
ISN 0176      IF(I2V(J)) 132,140,132
ISN 0177      132 NODE1=I2V(J)
ISN 0178      NODE2=I3V(J)
ISN 0179      IF(C(NODE1)) 136,134,136
ISN 0180      134 TR(NODE1)=TR(NODE1)+COND(J)*T(NODE2)
ISN 0181      136 IF(C(NODE2)) 140,138,140
ISN 0182      138 TR(NODE2)=TR(NODE2)+COND(J)*T(NODE1)
ISN 0183      140 CONTINUE
ISN 0184      RETURN
ISN 0185      1000 FORMAT(1X,3I5,E15.5,' ** GENERATED NODE **')
ISN 0186      1001 FORMAT(1X,I5,1GX,2E15.5,' ** GENERATED NODE **')
ISN 0187      1002 FORMAT(A4)
ISN 0188      1003 FORMAT(A1,I4,2I5,E15.5)
ISN 0189      2001 FORMAT(2GX,40('**'))/5X,'AT LEAST ONE OF THE RESISTORS HAS A NUMBER
                1 GREATER THAN 999'/5X,'CHECK THE INPUT AND MAKE THE NECESSARY CORRE
                2 CTIONS'
ISN 0190      2006 FORMAT(2GX,40('**'))/5X,'AT LEAST ONE OF THE NODES HAS A NUMBER GREA
                1 TER THAN 700'/5X,'CHECK THE INPUT AND MAKE THE NECESSARY CORRECTIO
                2 NS'
ISN 0191      9001 FORMAT(2GX,40('**'))/5X,'RESISTOR CR CCNDUCTOR',I4,' IS ATTACHED TO
                1A NODE ('',I3,'')/5X,'FOR WHICH NEITHER THE TEMPERATURE NOR THE CAP

```

2ACITANCE HAS BEEN SPECIFIED.)

9002 FORMAT(20X,40('*/5X,'THE TEMPERATURE AND CAPACITANCE FOR NODE',
114,' HAVE BEEN SPECIFIED,'/5X,'BUT THEY ARE NOT ATTACHED TO ANY RE

2SISTOR OR CONDUCTOR')

END

TRCI1900

ISN 0192

ISN 0193

COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF
SUBROUTINE FIN(T,I2V)

ISN 0002

C FIN

C THIS SUBROUTINE READS IN ALL FUNCTION DATA

C

FIN 1060

FIN 1070

FIN 1080

ISN 0003 COMMON/FUN1/ICATH,KC(50)

ISN 0004 COMMON/FUN2/IRAD,KR(250),EKR(250)

ISN 0005 COMMON/FUN3/IPHK,IPH(50),PHT1(50),PHT2(50),PHH2(50)

ISN 0006 COMMON/FUN4/ICONK,ICON(100),EXCON(100),ACON(100)

ISN 0007 COMMON/FUN5/IRB1,POWFAC(100),NODEQ(100),QO,IDEK

ISN 0008 COMMON/FUN6/IAER1,IAER2,IAERK,AER(51),IAER(50),I2AER(50),ALLOC(50)

ISN 0009 COMMON/FUN7/IKF,IAF2(250),IAF3(250),IAF4(250),FIAF5(250)

ISN 0010 COMMON/FUN8/IPK,IPR1(700),IPR2(700),PRUNT2(700)

ISN 0011 COMMON/FUN9/CO(20),PRTIME(5),TYM(5)

ISN 0012 COMMON/FUN10/KTHMAX,NEW(250),KLEAD(250),CELUID(250),KTABLE(30),

*LTABLE(250),CFMIN(30),FULTIM(250)

ISN 0013 COMMON/FUN11/JOINMX,JTABLE(50),NUNODE(50),INNODE(50)

ISN 0014 COMMON/FUN12/MIX,KKTABL(30),LLEAD(200),LLTABL(200),LNEW(200),

JCSOLID(200)

FIN 1140

&IDEADB

FIN 1150

FIN 1160

FIN 1170

ISN 0016 COMMON/FUN14/F14,ERROR,NUTIME,ADTIME(20)

ISN 0017 COMMON/VARY/ NODVAR,IRVAR(999)

ISN 0018 INTEGER*2 KC,KR,IPH,ICON,NODEQ,IAFR,I2AER,IAF2,IAF3,IAF4,IPR1,

JIPR2,NEW,KLEAD,KTABLE,LTABLE,JTABLE,NUNODE,INNODE,KKTABL,LLEAD,

2LLTABL,LNEW

ISN 0019 INTEGER*2 IRVAR,HOL,PRUNT2,I2V

ISN 0020 REAL*8 T

ISN 0021 COMMON/CRASH/SIGNAL

ISN 0022 DIMENSION I(3),HOL(4)

ISN 0023 DIMENSION T(700),I2V(999)

ISN 0024 EQUIVALENCE(I(1),I1),I(2),I2),I(3),I3)

ISN 0025 DATA HOL/8HK-T-C-Q-/

C

FORMAT STATEMENTS

ISN 0026

2001 FORMAT(2X,'THE NODES WITHIN THE RANGE OF THE LAST TWO CARDS CONTAI

IN THE SAME INFORMATION AS THESE CARDS')

```

ISN 0027 2002 FORMAT(2X,'NODES',I4,' TO',I4,' HAVE BEEN GENERATED INTERNALLY.');//
      12X,'THE INFORMATION IS THE SAME AS SHOWN FOR NODE',I4)
ISN 0028 9003 FORMAT(20X,40('**'))/5X,'YOU HAVE LESS THAN FOUR PIECES OF INFORMATI
      10N FOR PHASE TRANSITION AT ONE NODE')
ISN 0029 9006 FORMAT(20X,40('**'))/5X,'YOU HAVE ENTERED THE DATA FOR THE AERODYNAM
      11C HEATING ROUTINE INCORRECTLY')
ISN 0030 9010 FORMAT(20X,40('**'))/5X,'YOU HAVE USED ZERO AS ONE OF THE TABLE NUMB
      1ERS')
ISN 0031 9011 FORMAT(20X,40('**'))/5X,'YOU HAVE FORGOTTEN TO SPECIFY ANY OUTPUT')
ISN 0032 9012 FORMAT(20X,40('**'))/5X,'YOU HAVE NEGLECTED TO SPECIFY A PRINT INTER
      1VAL')
ISN 0033 9013 FORMAT(20X,40('**'))/5X,'YOU HAVE FORGOTTEN TO SPECIFY A PROBLEM COM
      1PLETION TIME')
ISN 0034 9100 FORMAT(20X,40('**'))/5X,'YOU ARE CALLING FOR A NON-EXISTENT FUNCTION
      1 NUMBER',I3)
ISN 0035 9110 FORMAT(20X,40('**'))/5X,'NODE',I4,' SPECIFIED IN FUNCTION',I3,' IS N
      1ON-EXISTENT')
ISN 0036 9111 FORMAT(20X,40('**'))/5X,'RESISTOR OR CONDUCTOR',I4,' SPECIFIED IN FU
      1NCTION',I3,' IS NON-EXISTENT')
ISN 0037 9115 FORMAT(20X,40('**'))/5X,'EITHER NODE',I4,' CR',I4,' SPECIFIED IN FUNC
      1TION',I3,' IS NON-EXISTENT')
ISN 0038 9150 FORMAT(20X,40('**'))/5X,'YOU HAVE USED MORE PIECES OF DATA IN FUNCTI
      1ON',I3,' THAN THE PROGRAM ALLOWS.//5X,'CHECK TO SEE THAT YOU DID N
      2OT OMIT A -1 AT THE END OF A PREVIOUS FUNCTION')

```

C

```

ISN 0039 00 5 J=1,999
ISN 0040 5 IRVAR(J)=0
ISN 0041 ICATH=0
ISN 0042 IRAD=0
ISN 0043 IPHK=0
ISN 0044 ICONK=0
ISN 0045 IRB1=0
ISN 0046 IAERK=0
ISN 0047 IKF=0
ISN 0048 IPK=0
ISN 0049 NDEADR=0
ISN 0050 KTHMAX=0
ISN 0051 JOINMX=0
ISN 0052 MIX=0
      TAP12190
      TAP12200
      TAP12210
      TAP12220
      TAP12230
      TAP12260
      TAP12270
      TAP12280
      TAP12310
      TAP12320
      TAP12330

```

ISN 0053	ISTEP=0	FIN 1340
ISN 0054	NSTEP=0	FIN 1350
ISN 0055	PRMAX=0.	FIN 1360
ISN 0056	TIMMAX=0.	FIN 1370
ISN 0057	FL4=0.	FIN 1380
ISN 0058	NUTIME=0	FIN 1390
ISN 0059	LOGICAL SIGNAL	FIN 1400

ISN 0060	1 CALL READ(I,FD,FD2)	
ISN 0061	IF(I1) 500,500,2	
ISN 0062	2 IF(I1-1) 10,10,19	

C		FIN 1440
C	C READ IN CATHODE FOLLOWER DATA, FUNCTION 1	FIN 1450
C		FIN 1460

ISN 0063	10 IFUN=1	
ISN 0064	KC(ICATH+1)=I2	
ISN 0065	KC(ICATH+2)=I3	
ISN 0066	IF(I(I2).NE.0.) GO TO 11	FIN 1480
ISN 0068	WRITE(6,9110) I2,IFUN	
ISN 0069	SIGNAL=.FALSE.	

ISN 0070	11 IF(I(I3).NE.0.) GO TO 12	
ISN 0072	WRITE(6,9110) I3,IFUN	
ISN 0073	SIGNAL=.FALSE.	
ISN 0074	12 ICATH=ICATH+2	

ISN 0075	IF(ICATH.GT.52) GO TO 515	
ISN 0077	CALL READ (I,FD)	FIN 1500

ISN 0078	IF(I1) 1,10,2	
ISN 0079	19 IF(I1-2) 20,20,29	

C		FIN 1530
C	C READ IN RADIATION RESISTOR DATA, FUNCTION 2	FIN 1540
C		FIN 1550

ISN 0080	20 IFUN=2	
ISN 0081	IF(I3) 23,21,21	
ISN 0082	21 IRAD=IRAD+1	
ISN 0083	IF(IRAD.GT.250) GO TO 515	
ISN 0085	KR(IRAD)=I3	FIN 1570

ISN 0086	IF(I2V(I3).NE.0) GO TO 22	
ISN 0088	WRITE(6,9111) I3,IFUN	
ISN 0089	SIGNAL=.FALSE.	
ISN 0090	22 EKR(IRAD)=FD	

ISN 0091	IRVAR(I3)=I3		
ISN 0092	CALL READ (I,FD)		FIN 1590
ISN 0093	GO TO 28		
ISN 0094	23 INIT=-I3		
ISN 0095	CALL READ(I,FD)		
ISN 0096	LAST=I3		
ISN 0097	DO 25 J=INIT, LAST		
ISN 0098	IRAD=IRAD+1		
ISN 0099	IF(IRAD.GT.250) GO TO 515		
ISN 0101	IRVAR(J)=J		
ISN 0102	KR(IRAD)=J		
ISN 0103	IF(I2V(J).NE.0) GO TO 25		
ISN 0105	WRITE(6,9111) J,IEUN		
ISN 0106	SIGNAL=.FALSE.		
ISN 0107	25 EKR(IRAD)=ED		
ISN 0108	WRITE(6,2002) INIT, LAST, LAST		
ISN 0109	28 IF(I1) 1,20,2		
ISN 0110	29 IF(I1-3) 30,30,39		FIN 1620
	C		FIN 1630
	C READ IN LATENT HEAT DATA, FUNCTION 3		FIN 1640
	C		
ISN 0111	30 IFUN=3		
ISN 0112	IPHK=IPHK+1		
ISN 0113	IF(IPHK.GT.50) GO TO 515		
ISN 0115	IPH(IPHK)=I3		FIN 1660
ISN 0116	PH1(IPHK)=FD		FIN 1670
ISN 0117	PH1(IPHK)=FD2		
ISN 0118	IF(T(I3).NE.0.) GO TO 31		
ISN 0120	WRITE(6,9110) I3, IFUN		
ISN 0121	SIGNAL=.FALSE.		
ISN 0122	31 CALL READ2 (I,PH12(IPHK),PH12(IPHK))		
ISN 0123	IF(I1) 33,35,33		
ISN 0124	33 WRITE (6,9003)		
ISN 0125	GO TO 999		FIN 1770
ISN 0126	35 CALL READ2 (I,FD,FD2)		
ISN 0127	IF(I1) 1,30,2		
ISN 0128	39 IF(I1-4) 40,40,49		
	C		FIN 1810
	C READ IN CONVECTION RESISTOR DATA, FUNCTION 4		FIN 1820

FIN 1830

C

```
ISN 0129      40 IFUN=4
ISN 0130      ICONK=ICONK+1
ISN 0131      IF (ICONK.GT.100) GO TO 515
ISN 0133      ICON (ICONK)=I3
ISN 0134      IF (I2V(I3).NE.0) GO TO 41
ISN 0136      WRITE(6,9111) I3,IFUN
ISN 0137      SIGNAL=.FALSE.
ISN 0138      41 EXCON(ICONK)=FD
ISN 0139      ACON(ICONK)=FD2
ISN 0140      IRVAR(I3)=I3
ISN 0141      CALL READ2 (I,FD,FD2)
ISN 0142      IF (I1) 1,40,2
ISN 0143      49 IF (I1-5) 50,50,59
```

FIN 1960

FIN 1970

FIN 1980

C READ IN ARBITRARY HEAT INPUT DATA, FUNCTION 5

C

```
ISN 0144      50 IFUN=5
ISN 0145      QO=FD
ISN 0146      51 CALL READ(I,FD)
ISN 0147      IF (I1.II.0) GO TO 1
ISN 0149      IF (I3) 52,53,52
ISN 0150      52 IRB1=IRB1+1
ISN 0151      IF (IRB1.GT.100) GO TO 515
ISN 0153      NDEQ(IRB1)=I3
ISN 0154      POWFAC(IRB1)=FD
ISN 0155      IF (I(I3).NE.0.) GO TO 51
ISN 0157      WRITE(6,9110) I3,IFUN
ISN 0158      SIGNAL=.FALSE.
ISN 0159      GO TO 51
ISN 0160      53 IDELK=I1
ISN 0161      GO TO 51
ISN 0162      55 IF (I1-6) 60,60,69
```

FIN 2040

FIN 2050

FIN 2110

FIN 2120

FIN 2130

C READ IN AERO HEATING DATA, FUNCTION 6

C

```
ISN 0163      60 IFUN=6
ISN 0164      IAER1=I2
ISN 0165      IAER2=I3
```

FIN 2150

ISN 0166	61 IAERK=IAERK+1	
ISN 0167	IF(IAERK.GT.50) GO TO 515	
ISN 0169	CALL READ(I,FD)	FIN 2170
ISN 0170	AER(IAERK)=FD	FIN 2180
ISN 0171	IAER(IAERK)=I3	FIN 2190
ISN 0172	IF(T(I3).NE.0.) GO TO 62	
ISN 0174	WRITE(6,9111) I3,IEUN	
ISN 0175	SIGNAL=.FALSE.	
ISN 0176	62 IF(I1) 1,63,2	
ISN 0177	63 IF(IAER2) 61,65,61	
ISN 0178	65 CALL READ(I(1),I(2),I2AER(IAERK),ALLOC(IAERK))	
ISN 0179	IF(I1) 67,61,67	
ISN 0180	67 WRITE (6,9006)	
ISN 0181	GO TO 999	FIN 2270
ISN 0182	69 IF(I1-7) 70,70,79	
	C	
	C READ IN TABULAR FUNCTION DATA, FUNCTION 7	FIN 2320
	C	FIN 2330
		FIN 2340
ISN 0183	70 IEUN=7	
ISN 0184	CALL READ (I,FD)	
ISN 0185	IF(I1) 1,1,71	
ISN 0186	71 IKF=IKF+1	
ISN 0187	IF(IKF.GT.250) GO TO 515	
ISN 0189	IAF2(IKF)=I1	FIN 2380
ISN 0190	IAF3(IKF)=I2	FIN 2390
ISN 0191	IAF4(IKF)=I3	FIN 2400
ISN 0192	IAF5(IKF)=FD	FIN 2410
ISN 0193	GO TO 70	
ISN 0194	79 IF(I1-8) 80,80,89	
	C	FIN 2440
	C READ IN PRINT SPECIFICATION, FUNCTION 8	FIN 2450
	C	FIN 2460
ISN 0195	80 IEUN=8	
ISN 0196	IF(I2) 83,81,81	
ISN 0197	81 IPK=IPK+1	
ISN 0198	IF(IPK.GT.700) GO TO 515	
ISN 0200	IPR1(IPK)=I2	
ISN 0201	IPR2(IPK)=I3	FIN 2490
ISN 0202	PRUNT2(IPK)=HOL(I2)	FIN 2500

ISN 0203 82 CALL READ(I,FD)

ISN 0204 IF(I1) 1,80,2

ISN 0205 83 INIT=-I3

ISN 0206 CALL READ(I,FD)

ISN 0207 LAST=I3

ISN 0208 WRITE (6,2002) INIT, LAST, LAST

ISN 0209 DO 84 J=INIT, LAST

ISN 0210 IPK=IPK+1

ISN 0211 IF(IPK.GT.700) GO TO 515

ISN 0213 IPR1(IPK)=I2

ISN 0214 IPR2(IPK)=J

ISN 0215 PRUNT2(IPK)=HOL(I2)

ISN 0216 84 CONTINUE

ISN 0217 GO TO 82

ISN 0218 85 IF(I1-9) 90,90,99

C

C READ IN PROBLEM CONSTANTS, FUNCTION 9

C

ISN 0219 90 IF(FD.EQ.0.) CALL READ2 (I,FD,FD2)

ISN 0221 M=I3

ISN 0222 IF(M.EQ. 1) GO TO 93

ISN 0224 91 IF(M.EQ. 2) GO TO 95

ISN 0226 CO(M)=FD

ISN 0227 CALL READ2(I,FD,FD2)

ISN 0228 92 IF(I1) 97,90,2

ISN 0229 93 CO(1)=1.

ISN 0230 94 CO(ISTEP+16)=FD

ISN 0231 ISTEP=ISTEP+1

ISN 0232 PRIME(ISTEP)=FD2

ISN 0233 IF(FD2.EQ.0.) PRIME(ISTEP)=CO(4)+1.

ISN 0235 IF(PRIME(ISTEP).GT.PRMAX) PRMAX=PRIME(ISTEP)

ISN 0237 CALL READ2 (I,FD,FD2)

ISN 0238 M=I3

ISN 0239 IF(M-1) 91,94,91

ISN 0240 95 CO(2)=1.

ISN 0241 96 CO(NSTEP+11)=FD

ISN 0242 NSTEP=NSTEP+1

ISN 0243 TYM(NSTEP)=FD2

ISN 0244 IF(FD2.EQ.0.) TYM(NSTEP)=CO(4)+1.

FIN 2540

FIN 2550

FIN 2560

FIN 2600

FIN 2690

FIN 2700

FIN 2710

FIN 2660

FIN 2790

FIN 2800

ISN 0246	IF(TYM(NSSTEP).GT.TIMMAX) TIMMAX=TYM(NSSTEP)	FIN 2810
ISN 0248	CALL READ2 (I,FD,FD2)	
ISN 0249	M=I3	FIN 2760
ISN 0250	IF(I1.LT.O) GO TO 97	
ISN 0252	IF(M-2) 52,96,92	
ISN 0253	97 IF(PRMAX.LT.CO(4)) PRTIME(ISTEP)=CO(4)+1.	
ISN 0255	IF(TIMMAX.II.CO(4)) TYM(NSSTEP)=CO(4)+1.	
ISN 0257	GO TO 1	
ISN 0258	99 IF(I1-10) 100,100,109	FIN 2950
	C	FIN 2960
	C READ IN TIME DELAY CATHODE FOLLOWER - FUNCTION 10	FIN 2970
	C	
ISN 0259	100 IFUN=10	
ISN 0260	CALL READ2 (I,FD,FD2)	
ISN 0261	IF(I1) 1,107,101	
ISN 0262	101 IF(I3) 104,102,102	
ISN 0263	102 KTHMAX=KTHMAX+1	
ISN 0264	IF(KTHMAX.GT.250) GO TO 515	FIN 3010
ISN 0266	NEW(KTHMAX)=I2	FIN 3020
ISN 0267	KLEAD(KTHMAX)=I3	
ISN 0268	IF(T(I3).NE.O..AND.I(I2).NE.O.) GO TO 103	
ISN 0270	WRITE(6,9115) I2,I3,IFUN	
ISN 0271	SIGNAL=.FALSE.	
ISN 0272	103 KTABLE(I1)=I1	FIN 3040
ISN 0273	KTABLE(KTHMAX)=I1	FIN 3050
ISN 0274	CFLUID(KTHMAX)=FD	
ISN 0275	FULTIM(KTHMAX)=FD2	
ISN 0276	IF(CFLUID(KTHMAX).EQ.O.) GO TO 100	
ISN 0278	IF(CFLUID(KTHMAX).LT.CFMIN(I1)) CFMIN(I1)=CFLUID(KTHMAX)	FIN 3070
ISN 0280	GO TO 100	
ISN 0281	104 INIT=-I3	
ISN 0282	NEWNOI=I2	
ISN 0283	KTABLE(I1)=I1	
ISN 0284	CALL READ2(I,FD,FD2)	
ISN 0285	LAST=I3	
ISN 0286	JSAVE=0	
ISN 0287	DO 106 J=INIT,LAST	
ISN 0288	KTHMAX=KTHMAX+1	
ISN 0289	IF(KTHMAX.GT.250) GO TO 515	

ISN 0291	KFOLOW=NEWNOD+JSAVE	
ISN 0292	NEW(KTHMAX)=KFOLOW	
ISN 0293	KLEAD(KTHMAX)=J	
ISN 0294	IF(T(J).NE.O..AND.T(KFOLOW).NE.O.) GO TO 105	
ISN 0296	WRITE(6,9115) KFOLOW,J,IFUN	
ISN 0297	SIGNAL=.FALSE.	
ISN 0298	105 LTABLE(KTHMAX)=I1	
ISN 0299	CFLUID(KTHMAX)=FD	
ISN 0300	FULTIM(KTHMAX)=FD2	
ISN 0301	IF(J.GT.INIT) GO TO 106	
ISN 0303	IE(CFLUID(KTHMAX).EQ.O) GO TO 106	
ISN 0305	IF(CFLUID(KTHMAX).LT.CFMIN(I1)) CFMIN(I1)=CFLUID(KTHMAX)	
ISN 0307	106 JSAVE=JSAVE+1	
ISN 0308	WRITE(6,2001)	
ISN 0309	GO TO 100	
ISN 0310	107 WRITE(6,9010)	
ISN 0311	GO TO 999	
ISN 0312	109 IF(I1-I1) 110,110,119	FIN 3120
	C	
	C TEMPERATURE AVERAGING SCHEME - FUNCTION 11	FIN 3130
	C	FIN 3140
ISN 0313	110 IFUN=11	
ISN 0314	CALL READ(I,FD)	
ISN 0315	IF(I1) 1,111,111	
ISN 0316	111 JCINMX=JCINMX+1	
ISN 0317	IF(JCINMX.GT.50) GO TO 515	
ISN 0319	JTABLE(JCINMX)=I1	FIN 3190
ISN 0320	NUNODE(JCINMX)=I2	FIN 3200
ISN 0321	INNODE(JCINMX)=I3	FIN 3210
ISN 0322	IF(T(I2).NE.O..AND.T(I3).NE.O.) GO TO 110	
ISN 0324	WRITE(6,9115) I2,I3,IFUN	
ISN 0325	SIGNAL=.FALSE.	
ISN 0326	GO TO 110	
ISN 0327	119 IF(I1-I2) 120,120,129	FIN 3230
	C	
	C FUNCTION 12 TEMPERATURE ADVANCE THROUGH A SERIES OF NODES	FIN 3240
	C	FIN 3250
ISN 0328	120 IFUN=12	
ISN 0329	CALL READ(I,FD)	

ISN 0330	IF(I1) 1,107,121	
ISN 0331	121 IF(I3) 124,122,122	
ISN 0332	122 MIX=MIX+1	
ISN 0333	IF(MIX.GT.200) GO TO 515	
ISN 0335	LNEW(MIX)=I2	FIN 3300
ISN 0336	LLEAD(MIX)=I3	FIN 3310
ISN 0337	IF(T(I2).NE.0..AND.T(I3).NE.0.) GO TO 123	
ISN 0339	WRITE(6,9115) I2,I3,IFUN	
ISN 0340	SIGNAL=.FALSE.	
ISN 0341	123 LLTABL(MIX)=I1	FIN 3330
ISN 0342	KKTABL(I1)=I1	FIN 3340
ISN 0343	CSOLID(MIX)=FD	
ISN 0344	GO TO 120	
ISN 0345	124 INIT=-I3	
ISN 0346	NEWNOD=I2	
ISN 0347	KKTABL(I1)=I1	
ISN 0348	CALL READ(I,FD)	
ISN 0349	LAST=I3	
ISN 0350	JSAVE=0	
ISN 0351	DO 125 J=INIT, LAST	
ISN 0352	MIX=MIX+1	
ISN 0353	IF(MIX.GT.200) GO TO 515	
ISN 0355	LFOLOW=NEWNOD+JSAVE	
ISN 0356	LNEW(MIX)=LFOLOW	
ISN 0357	LLEAD(MIX)=J	
ISN 0358	LLTABL(MIX)=I1	
ISN 0359	CSOLID(MIX)=FD	
ISN 0360	IF(T(J).NE.0..AND.T(LFOLOW).NE.0.) GO TO 125	
ISN 0362	WRITE(6,9115) LFOLOW,J,IFUN	
ISN 0363	SIGNAL=.FALSE.	
ISN 0364	125 JSAVE=JSAVE+1	
ISN 0365	WRITE(6,2001)	
ISN 0366	GO TO 120	
ISN 0367	129 IF(I1-I3) 130,130,139	FIN 3360
	C	
	C FUNCTION 13 - TEMPERATURE FEEDBACK AND DEADBAND CONTROLLER INPUT	FIN 3370
	C	FIN 3380
ISN 0368	130 DO 131 J=1,3	
ISN 0369	CALL READ2 (I,FD,FD2)	FIN 3410

ISN 0370	NBASE(J)=I3	FIN 3420
ISN 0371	RHO(J)=FD	FIN 3430
ISN 0372	131 TRASE(J)=FD2	
ISN 0373	CALL READ2 (I,FD,FD2)	FIN 3450
ISN 0374	IF(I3) 1,1,132	
ISN 0375	132 DBWAIT=FD2	
ISN 0376	NDEADR=I3	FIN 3480
ISN 0377	DELKDB=FD	FIN 3490
ISN 0378	CALL READ2 (I,FD,FD2)	FIN 3500
ISN 0379	DELTDR=FD2	FIN 3510
ISN 0380	TDEADR=FD	FIN 3520
ISN 0381	CALL READ(I,FD)	FIN 3530
ISN 0382	GO TO 1	
ISN 0383	139 IF(I1-14) 140,140,510	

C

C FUNCTION 14 - STEADY STATE OPTION

C

ISN 0384	140 F14=1.	
ISN 0385	ERROR=ED	FIN 3570
ISN 0386	141 CALL READ(I,FD)	
ISN 0387	IF(I1) 1,142,142	
ISN 0388	142 NUTIME=11	
ISN 0389	ADTIME(I1)=ED	FIN 3610
ISN 0390	GO TO 141	
ISN 0391	500 IF(IPK) 501,501,502	
ISN 0392	501 WRITE (6,9011)	
ISN 0393	SIGNAL=.FALSE.	
ISN 0394	502 IF(CO(16)) 503,503,504	
ISN 0395	503 WRITE (6,9012)	
ISN 0396	SIGNAL=.FALSE.	
ISN 0397	504 IF(CO(4)) 505,505,506	
ISN 0398	505 WRITE (6,9013)	
ISN 0399	SIGNAL=.FALSE.	
ISN 0400	506 RETURN	
ISN 0401	510 WRITE (6,9100) 11	
ISN 0402	SIGNAL=.FALSE.	FIN 3650
ISN 0403	RETURN	FIN 3670
ISN 0404	515 WRITE (6,9150) IFUN	
ISN 0405	999 CONTINUE	

ISN 0406

ISN 0407

ISN 0408

SIGNAL = FALSE.

RETURN

END

FIN 3740

```

      COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF
ISN 0002      SUBROUTINE FLOW(TOME,WCP,T2ME,TABLE,DELTAT,T,C,RI,RCMIN,IERR)
ISN 0003      REAL*8 I,C,TNEW,TEMP,Q,IR,RI
ISN 0004      INTEGER*2 NEW,KLEAD,KTABLE,LTABLE,KKTABL,LLEAD,LLTABL,LNEW
ISN 0005      COMMON/EUN10/KTHMAX,NEW(250),KLEAD(250),CFLUID(250),KTABLE(30),
      *LTABLE(250),CFMIN(30),FULTIM(250)
ISN 0006      COMMON/EUN12/MIX,KKTABL(30),LLEAD(200),LLTABL(200),LNEW(200),
      1CSOLID(200)
ISN 0007      COMMON/CRASH/SIGNAL
ISN 0008      DIMENSION WCP(30),TABLE(104,30),T(700),C(700),TNEW(250),WCPDT(30),
      1TEMP(200),Q(700),IR(700),RI(700)
ISN 0009      LOGICAL SIGNAL
ISN 0010      IFUN=10
ISN 0011      KERR10=0
ISN 0012      IF(KEYIC.NE.1) GO TO 5
ISN 0014      KERR10=1
      C
      C CALCULATE WCP AND RESET DELTAT IF NECESSARY
      C
ISN 0015      5 DO 10 I=1,30
ISN 0016      IF(KTABLE(I).EQ. 0) GO TO 10
ISN 0018      INDEX=KTABLE(I)
ISN 0019      MC=(INDEX-1)*104+1
ISN 0020      WCP(I)=ENTERP(T2ME,TABLE(MC,1))
ISN 0021      IF(.NOT. SIGNAL) RETURN
ISN 0023      IF(WCP(I).EQ. 0.) GO TO 9
ISN 0025      IF(CFMIN(I).EQ. 0.)GO TO 9
ISN 0027      IF(CFMIN(I)/WCP(I).LT.DELTAT) DELTAT=CFMIN(I)/WCP(I)
ISN 0029      9 WCPDT(I)=WCP(I)*DELTAT
ISN 0030      10 CONTINUE
ISN 0031      DO 20 I=1,KTHMAX
ISN 0032      KNEW=KLEAD(I)
ISN 0033      20 TNEW(I)=T(KNEW)
ISN 0034      DO 40 I=1,KTHMAX
ISN 0035      KTHNEW=NEW(I)
ISN 0036      J=LTABLE(I)
ISN 0037      IF(T2ME.LT.FULTIM(I)) GO TO 40
ISN 0039      IF(C(KTHNEW).LT. 0.) C(KTHNEW)=CFLUID(I)

```



```

ISN 0041      IF(CFLUID(I).EQ. 0.) GO TO 30
ISN 0043      T(KTHNEW)=(T(KTHNEW)*(C(KTHNEW)-WCPDT(J))+TNEW(I))*WCPDT(J))
               *ZC(KTHNEW)
ISN 0044      GO TO 40
ISN 0045      30 T(KTHNEW)=TNEW(I)
ISN 0046      IF(WCP(J).EQ. 0.) GO TO 40
ISN 0048      C(KTHNEW)=WCPDT(I)
ISN 0049      IF(C(KTHNEW)/RI(KTHNEW).GE.RCMIN) GO TO 40
ISN 0051      IF(DELIAT.GE.1.0) GO TO 40
ISN 0053      IF(KEY10.NE.1) GO TO 38
ISN 0055      WRITE(6,9000) IFUN,KTHNEW
ISN 0056      38 IERR=1
ISN 0057      40 CONTINUE
ISN 0058      KEY10=IERR
ISN 0059      IF(KERR10.NE.1) IERR=0
ISN 0061      RETURN
ISN 0062      ENTRY ESTFLO(TOME,WCP,T2ME,TABLE,DELIAT,I,C,Q,IR,RI,RCMIN,IERR)
ISN 0063      IF(TOME.NE. 0.) GO TO 52
ISN 0065      DO 51 I=1,MIX
ISN 0066      N=LNEW(I)
ISN 0067      51 TEMP(I)=T(N)
ISN 0068      52 IFUN=12
ISN 0069      KERR12=0
ISN 0070      IF(KEY12.NE.1) GO TO 55
ISN 0072      KERR12=1
ISN 0073      55 DO 60 I=1,30
ISN 0074      IF(KKTABL(I).EQ. 0) GO TO 60
ISN 0076      INDEX=KKTABL(I)
ISN 0077      MC=(INDEX-1)*104+1
ISN 0078      WCP(I)=ENTERP(T2ME,TABLE(MC,1))
ISN 0079      WCPDT(I)=WCP(I)*DELIAT
ISN 0080      IF(.NOT. SIGNAL) RETURN
ISN 0082      60 CONTINUE
ISN 0083      DO 70 I=1,MIX
ISN 0084      N=LNEW(I)
ISN 0085      J=LLTABL(I)
ISN 0086      L=LLLEAD(I)
ISN 0087      IF(WCP(J).EQ. 0.) GO TO 70
ISN 0089      C(N)=CSOLID(I)+WCPDT(J)

```

```

ISN 0090      IF(C(N)/RI(N).GE.RCMIN) GO TO 65
ISN 0092      IF(DELTAT,GE.1,0) GO TO 65
ISN 0094      IF(KEY12,NE.1) GO TO 63
ISN 0096      WRITE(6,9000)IFUN,N
ISN 0097      63 IERR=1
ISN 0098      65 TEMP(I)=(TEMP(I)*CSOLID(I)+WCPDT(J)*T(L))/C(N)
ISN 0099      TEMP(I)=DELTAT/C(N)*(Q(N)+TR(N)-TEMP(I)*RI(N))+TEMP(I)
ISN 0100      T(N)=TEMP(I)
ISN 0101      70 CONTINUE
ISN 0102      KEY12=IERR
ISN 0103      IF(KERR12,NE.1) IERR=0
ISN 0105      RETURN
ISN 0106      9000 FORMAT(1/5X,'THE PROBLEM IS UNSTABLE IN FUNCTION',I3/10X,'THE RC PR
            10DUCT FOR NODE',I4,' IS LESS THAN THE CURRENT RCMIN AND MUST BE CO
            2RRECTED')
ISN 0107      END

```

```
      COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NODEIT,ID,XREF
ISN 0002      SUBROUTINE HEATIN(T2ME, TABLE, QBETA, T, Q, DBTOT)
ISN 0003      INTEGER*2 NDEQ
ISN 0004      COMMON/FUN5/IRB1,POWFAC(100),NODEQ(100),QO,IDEK
ISN 0005      COMMON/FUN13/NBASE(3),RHO(3),TBASE(3),DBWAIT,NDEADB,DELKCB,DELIDB,T171112
      &TDEADB
ISN 0006      COMMON/CRASH/SIGNAL
ISN 0007      EQUIVALENCE (NBASE1,NBASE(1)),(NBASE2,NBASE(2)),(NBASE3,NBASE(3))
ISN 0008      LOGICAL SIGNAL
ISN 0009      REAL*8 T,Q,QBETA
ISN 0010      DIMENSION TABLE(104,30),QBETA(100),T(700),Q(700)
ISN 0011      MC=(IDEK-1)*104+1
ISN 0012      DELKC=ENTERP(T2ME, TABLE(MC,1))
ISN 0013      IF(.NOT. SIGNAL) RETURN
      C
      C      DEAD BAND CONTROLLER
      C
ISN 0015      IF(NDEADB) 10,10,1
ISN 0016      1 IF(DABS(T(NDEADB)-TDEADB),T,DELIDB) GO TO 10
ISN 0018      IF(T2ME-DBGC) 10,5,5
ISN 0019      5 DBGC=T2ME+DBWAIT
ISN 0020      IF(T(NDEADB)-TDEADB) 6,10,7
ISN 0021      6 DBTOT=DBTOT+DELKCB
ISN 0022      GO TO 10
ISN 0023      7 DBTOT=DBTOT-DELKCB
      C
      C      RESUME THE HEATING CALCULATIONS
      C
ISN 0024      10 DELK=RHO(1)*(T(NBASE1)-TBASE(1))+RHO(2)*(T(NBASE2)-TBASE(2))
      &+RHO(3)*(T(NBASE3)-TBASE(3))+DELKCB+DBTOT
ISN 0025      DELRHO=.0077-DELK
ISN 0026      IF(DELRHO.LE. 0.) GO TO 910
ISN 0028      DELEXP=.10044*DELK*T2ME/DELRHO
ISN 0029      RHOEXP=DELRHO*T2ME/8.1E-6
ISN 0030      IF(DELEXP.LT.-100.) GO TO 15
ISN 0032      EX1=EXP(DELEXP)
ISN 0033      GO TO 17
ISN 0034      15 EX1=0.
```

```

ISN 0035      17 IF(RHOEXP.GT. 150.) GO TO 19
ISN 0037      IF(RHOEXP.GT. 0.) EX2=1./EXP(RHCEXP)
ISN 0039      GO TO 25
ISN 0040      19 EX2=0.
ISN 0041      25 RCONST=Q0*(.0077*EX1-DELK*EX2)/DEL RHO
ISN 0042      DO 30 I=1,IRB1
ISN 0043      NODQ=NODEQ(I)
ISN 0044      Q(NODQ)=POWFAC(I)*RCONST
ISN 0045      IF(T2ME.EQ. 0.) RETURN
ISN 0047      QBETA(I)=6.22E-2/(T2ME**.2)*Q0*POWFAC(I)
ISN 0048      IF(Q(NODQ)*.06 .LT. QBETA(I)) Q(NODQ)=Q(NODQ)*.94+QBETA(I)
ISN 0050      30 CONTINUE
ISN 0051      RETURN
ISN 0052      910 WRITE (6,9010) DELK
ISN 0053      9010 FORMAT(1H0,40THE REACTOR IS PROMPT CRITICAL, DELTA K=,1PE10.5) T
ISN 0054      SIGNAL=.FALSE.
ISN 0055      RETURN
ISN 0056      END

```

TAP14450

TAP14480

TAP14490

```

COMPILER OPTIONS - NAME= TAP7,OPI=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF
ISN 0002      SUBROUTINETABIN(TABLE,KDEP,KIND)
               C TABIN
               C
               C THIS ROUTINE READS IN THE TABULAR DATA
               C
ISN 0003      COMMON/EUN7(IKE,IAF2(250),IAF3(250),IAF4(250),FIAF5(250))
ISN 0004      COMMON/VARY/ NODVAR,IRVAR(999)
ISN 0005      INTEGER*2 IRVAR
ISN 0006      INTEGER*2 KIND,KDEP
ISN 0007      DIMENSION TABLE(104,30),KDEP(30),KIND(30)
ISN 0008      COMMON/CRASH/SIGNAL
ISN 0009      LOGICAL SIGNAL
ISN 0010      DIMENSION ITAB(3),I(3)
ISN 0011      EQUIVALENCE (ITAB(1),NTAB),(ITAB(2),NIND),(ITAB(3),NDEP),
               *(I(1),N1)
ISN 0012      101 CALL READ2 (ITAB,PER,CONST)
ISN 0013      IF(NTAB) 120,120,105
ISN 0014      105 IF(NDEP,NE,1) GO TO 109
ISN 0016      DO 108 J=1,IKF
ISN 0017      IF(NTAB-IAF4(J)) 108,106,108
ISN 0018      106 NODVAR=NODVAR+1
ISN 0019      K=IAF2(J)
ISN 0020      IRVAR(K)=IAF2(J)
ISN 0021      108 CONTINUE
ISN 0022      109 TABLE (1,NTAB)=PER
ISN 0023      KDEP(NTAB)=NDEP
ISN 0024      TABLE(2,NTAB)=CCNST
ISN 0025      KIND(NTAB)=NIND
ISN 0026      N1=1
ISN 0027      DO 110J=3,103,2
ISN 0028      CALL READ2 (I,NTAB),TABLE(J+1,NTAB))
ISN 0029      IF(N1)101,110,110
ISN 0030      110 CONTINUE
ISN 0031      WRITE (6,111)NTAB
ISN 0032      111 FORMAT(103H0YOU HAVE FORGOTTEN THE MINUS ONE AT THE END OF THE PRETAB11240
               1VIOUS TABLE OR USED MORE THAN 50 POINTS CN TABLE I2)
ISN 0033      SIGNAL=.FALSE.

```

ISN 0034
ISN 0035

120 RETURN
END

TABU1280

COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41, SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF

```

ISN 0002      SUBROUTINE TAVG(TOME,C,T,WCP,DELTAT,RI,RCMIN,IERR)
ISN 0003      REAL*8 CSUM,CTSUM,T,C,RI
ISN 0004      INTEGER*2 JTABLE,NUNODE,INNOD
ISN 0005      COMMON/FUN11/JOINMX,JTABLE(50),NUNODE(50),INNOD(50)
ISN 0006      DIMENSION C(700),T(700),WCP(30),RI(700)
ISN 0007      IF(TOME.EQ. 0.) GO TO 100
ISN 0009      IFUN=11
ISN 0010      CSUM=0.
ISN 0011      CTSUM=0.
ISN 0012      KERR11=0
ISN 0013      IF(KEY11.NE.1) GO TO 5
ISN 0015      KERR11=1
ISN 0016      5 DO 50 I=1,JOINMX
ISN 0017      INNOD=INNOD(I)
ISN 0018      NUNOD=NUNODE(I)
ISN 0019      JTABLE=JTABLE(I)
ISN 0020      IF(I.EQ.1) GO TO 10
ISN 0022      IF(NUNODE(I).NE.NUNODE(I-1)) GO TO 20
ISN 0024      10 CSUM=CSUM+WCP(JTABLE)*DELTAT
ISN 0025      CTSUM=CTSUM+WCP(JTABLE)*DELTAT*(INNOD)
ISN 0026      IF(I.EQ.JOINMX) GO TO 30
ISN 0028      GO TO 50
ISN 0029      20 NUNOD=NUNODE(I-1)
ISN 0030      30 IF(CSUM.EQ. 0.) GO TO 40
ISN 0032      C(NUNOD)=CSUM
ISN 0033      IF(C(NUNOD)/RI(NUNOD).GE.RCMIN) GO TO 35
ISN 0035      IF(DELTAT.GE.1.0) GO TO 35
ISN 0037      IF(KEY11.NE.1) GO TO 33
ISN 0039      WRITE(6,9000) IFUN,NUNOD
ISN 0040      32 IERR=1
ISN 0041      35 T(NUNOD)=CTSUM/CSUM
ISN 0042      40 CSUM=WCP(JTABLE)*DELTAT
ISN 0043      CTSUM=WCP(JTABLE)*DELTAT*T(INNOD)
ISN 0044      50 CONTINUE
ISN 0045      KEY11=IERR
ISN 0046      IF(KERR11.NE.1) IERR=0
ISN 0048      RETURN

```

ISN 0049

100 TIME=1.

ISN 0050

RETURN

ISN 0051

9000

FORMAT(15X,

'THE PROBLEM IS UNSTABLE IN FUNCTION',I3/I0X,'THE RC PR

DUCT FOR NODE',I4,' IS LESS THAN THE CURRENT RCMIN AND MUST BE CO

RRRECTED')

END

ISN 0052


```
COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF
ISN 0002 SUBROUTINE STEADY(KEY,T,MAXNO,DELTA) STEAL010
ISN 0003 COMMON/EUN14/E14,ERROR,NUTIME,ADTIME(20) STEAL020
ISN 0004 REAL*8 T, TSAVE STEAL030
ISN 0005 DIMENSION T(700),TSAVE(700)
ISN 0006 KEY=1 STEAL050
ISN 0007 DO 50 I=1,MAXNO STEAL060
ISN 0008 IF(T(I).EQ.0.) GO TO 50 STEAL070
ISN 0009 DELTA=DABS((T(I)-TSAVE(I))/(T(I)*DELTA)) STEAL080
ISN 0010 IF(DELTA.LE.ERROR) GO TO 50 STEAL090
ISN 0011 KEY=0 STEAL100
ISN 0012 GO TO 90 STEAL110
ISN 0013 50 CONTINUE STEAL120
ISN 0014 GO TO 101 STEAL130
ISN 0015 90 DO 100 I=1,MAXNO STEAL140
ISN 0016 100 TSAVE(I)=T(I) STEAL150
ISN 0017 101 RETURN STEAL160
ISN 0020 END STEAL170
```

```

COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF
TSN 0002 SUBROUTINE LAT (PHT1,PHH1,PHT2,PHH2,QDEL,T,TPREV,
          IQ,RI,TR,DELAT,TIME,K)
          C LAT
          C
          C ROUTINE FOR LATENT HEAT EFFECTS
          C
          REAL*8 I,RI,TR,Q,TPREV
          COMMON/CRASH/SIGNAL
          LOGICAL SIGNAL
          IF(QDEL)2,3,2
          2 IF(TPREV-PHT2)16,4,16
          4 TLH=PHT2
          IF(PHH2)30,20,31
          30 WRITE (6,22)TIME,K
          SIGNAL=.FALSE.
          RETURN
          32 FORMAT(1H0//40H PROBLEM CUT OFF BY LATENT HEAT ROUTINE.F12.2,I5)
          31 EMDLH=PHH2
          6 QCH=(TR+Q-TLH*RI)*DELAT+QDEL
          IF(ABS(QCH)-EMDLH)7,8,8
          8 WRITE (6,9)TIME,K,T
          9 FORMAT(1H0//24H LATENT HEAT INFORMATIONF12.2,I5,F10.2)
          QDEL=0.0
          1 TPREV=T
          RETURN
          2 IF(T-PHT2)10,29,11
          10 IF(TPREV-PHT2)12,1,13
          12 IF(T-PHT1)14,20,15
          14 IF(TPREV-PHT1)1,1,26
          26 WRITE(6,9)TIME,K,T
          16 TLH=PHT1
          EMDLH=PHH1
          GO TO 6
          15 IF(TPREV-PHT1)26,26,1
          13 WRITE (6,9)TIME,K,T
          GO TO 4
          11 IF(TPREV-PHT2)13,13,1

```

ISN 0034	7 IF(QCH)17,19,18	LAT 1380
ISN 0035	17 IF(QDEL)19,19,8	LAT 1390
ISN 0036	19 QDEL=QCH	LAT 1400
ISN 0037	T=TLH	LAT 1410
ISN 0038	GO TO 1	LAT 1420
ISN 0039	18 IF(QDEL)8,19,19	LAT 1430
ISN 0040	29 IF(TPREV-PHI2)8,1,8	LAT 1440
ISN 0041	20 IF(TPREV-PHI1)8,1,8	LAT 1450
ISN 0042	END	LAT 1460

COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF

ISN 0002 FUNCTION ENTERP(ARG,TAB)

C ENTERP

ENTE1010

ENTE1020

C

ENTE1030

ENTE1040

C LINEAR INTERPOLATION ROUTINE

ENTE1050

C

COMMON/CRASH/SIGNAL

ENTE1070

ENTE1080

COMMON/TABNO/INDEX

ENTE1090

LOGICAL SIGNAL

ENTE1100

DIMENSION TAB(104)

ENTE1120

IF(TAB(2)) 1,2,1

ENTE1110

ENTE1120

1 ENTERP=TAB(2)

ENTE1170

ENTE1180

ENTE1190

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

ENTE1210

COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF

ENTE1340

BLOCK DATA
COMMON/CRASH/ SIGNAL

ENTE1360

LOGICAL SIGNAL

ENTE1370

END

COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF

ISN 0002	C	SUBROUTINE READ (I,X)	READ1030
ISN 0003	C	COMMON/FLAG/ KFLAG	READ1040
ISN 0004	C	DATA CHECK/'C'/	READ1050
ISN 0005		DIMENSION I(3)	READ1060
ISN 0006		READ (5,1002) C,I,X	
ISN 0007		IF(CHECK.EQ.C) GO TO 5	
ISN 0009		4 WRITE(6,2001) I,X	
ISN 0010		RETURN	
ISN 0011		5 KFLAG=0	
ISN 0012		WRITE(6,2002) I,X	
ISN 0013		RETURN	
ISN 0014		1002 FORMAT(A1,I4,2I5,E15.5,50H	
ISN 0015		2001 FORMAT(1X,3I5,E15.5,50H	
ISN 0016		2002 FORMAT(1X,'*',I4,2I5,E15.5,50H	
ISN 0017	C	ENTRY READ2(I,X,Y)	READ1020
	C	C READ ROUTINE TO READ IN DATA WITH TWO E15.5 FIELDS	READ1030
	C		READ1040
ISN 0018		READ (5,1003) I,X,Y	READ1060
ISN 0019		WRITE (6,2003) I,X,Y	
ISN 0020		RETURN	READ1080
ISN 0021		1003 FORMAT(3I5,2E15.5,35H	READ1090
ISN 0022		2003 FORMAT(1X,3I5,2E15.5,35H	
ISN 0023		END	READ1100

COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41, SOURCE,EBCDIC,NCLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF

ISN 0002

FUNCTION QECK (TIME,FLAG,T,TABH,TABM,TABA,TABR,TABREX,TABS

1,IBFLAG

QECK1010

QECK1020

2,I2AER,ALLOC

QECK1030

3,IABTLI,TABMLM,IABT,A,B,E,GAM,IABPR,EK)

QECK1040

C QECK

QECK1050

C

QECK1060

C

ECKERT EQUATION FOR AERODYNAMIC HEATING

QECK1070

C

QECK1080

ISN 0003

DIMENSION TABH(104),TABM(104),TABA(104),TABB(104),TABREX(104),

IABS(104),IABTLI(104),IABMLM(104),IABT(104),IABPR(104)

QECK1090

QECK1100

COMMON/CRASH/SIGNAL

LOGICAL SIGNAL

QECK1120

IF(FLAG)19,2,19

QECK1130

2 H=ENTERP(TIME,TABH)

QECK1140

IF(.NOT.SIGNAL) GO TO 7

QECK1150

EM=ENTERP(TIME,TABM)

QECK1160

IF(.NOT.SIGNAL) GO TO 7

QECK1170

IF(IBEFLAG)4,3,4

QECK1180

4 BETA=1.0

QECK1190

GO TO 5

QECK1200

3 ALPHA=ENTERP(TIME,TABA)

QECK1210

IF(.NOT.SIGNAL) GO TO 7

QECK1220

5 REXM=10.0*(ENTERP(H,TABREX))

QECK1230

IF(.NOT.SIGNAL) GO TO 7

QECK1240

6 S=ENTERP(EM,TABS)

QECK1250

IF(.NOT.SIGNAL) GO TO 7

QECK1260

9 TLOTI=ENTERP(EM,IABTLT)

QECK1270

IF(.NOT.SIGNAL) GO TO 7

QECK1280

EMLOMI=ENTERP(EM,TABMLM)

QECK1290

IF(.NOT.SIGNAL) GO TO 7

QECK1300

11 TINF=ENTERP(H,IABT)

QECK1310

IF(.NOT.SIGNAL) GO TO 7

QECK1320

EN=.71**E*(GAM-1.0)/2.0*(EMLOMI*EM)**2

QECK1330

TI=TLOTI*TINF

QECK1340

TR=TL*(1.0+EN)

QECK1350

FLAG=1.0

QECK1360

19 IF(IBEFLAG)1,20,1

QECK1370

ISN 0039	20	ALPHL=ALLOC-ALPH	QECK1380
ISN 0040		IF(I2AER)12,14,12	QECK1390
ISN 0041	12	ALPHI=-ALPHI	QECK1400
ISN 0042	14	IF(TABB(4))15,16,15	QECK1410
ISN 0043	15	BETA=ENTERP(ALPHI,TABB)	QECK1420
ISN 0044		IF(.NOT.SIGNAL) GO TO 7	QECK1430
ISN 0046		GO TO 1	QECK1440
ISN 0047	16	IF(ALPHI)17,18,18	QECK1450
ISN 0048	17	BETA=TABB(3)*ALPHI+1.0	QECK1460
ISN 0049		GO TO 1	QECK1470
ISN 0050	18	BETA=TABB(5)*ALPHI+1.0	QECK1480
ISN 0051	1	TLPOTL=.5*(1.0+(T+459.69)/TL)+.22*EN	QECK1490
ISN 0052		PR3K=10.0*(ENTERP(TLPOTL*TL,TARPR))	QECK1500
ISN 0053		IF(.NOT.SIGNAL) GO TO 7	QECK1510
ISN 0055		QECK=BETA*EK*(S*REXM*EM/TLPCTI**1.69)**A*EM	QECK1520
		1**B*PR3K*(TR-T-459.69)	QECK1530
ISN 0056	7	RETURN	QECK1540
ISN 0057		END	QECK1550


```

COMPILER OPTIONS - NAME= TAP7,OPT=02,LINECNT=41,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,NOEDIT,ID,XREF
FUNCTION QRW(TIME,FLAG,IRFLAG,T,TABH,TABM,TABA,A,B,E,GAM,EK,TABREXQRW 1010
1,I2AER,ALLOC QRW 1020
2,TABT,TABB) QRW 1030
C QRW QRW 1040
C QRW 1050
C R-W EQUATION FOR AERODYNAMIC HEATING QRW 1060
C QRW 1070
C DIMENSION TABH(104),TABM(104),TABA(104),TABB(104),TABREX(104), QRW 1080
ITABT(104) QRW 1090
COMMON/CRASH/SIGNAL
LOGICAL SIGNAL QRW 1110
IF(FLAG)9,2,19 QRW 1120
2 H=ENTERP(TIME,TABH) QRW 1130
IF(.NOT.SIGNAL) GO TO 7 QRW 1140
EM=ENTERP(TIME,TABM) QRW 1150
IF(.NOT.SIGNAL) GO TO 7 QRW 1160
IF(IBFLAG)4,3,4 QRW 1170
4 BETA=1.0 QRW 1180
GO TO 5 QRW 1190
3 ALPHA=ENTERP(TIME,TABA) QRW 1200
IF(.NOT.SIGNAL) GO TO 7 QRW 1210
5 REXM=10.0** (ENTERP(H,TABREX)) QRW 1220
IF(.NOT.SIGNAL) GO TO 7 QRW 1230
TINF=ENTERP(H,TABT) QRW 1240
IF(.NOT.SIGNAL) GO TO 7 QRW 1250
EN=.71**E*(GAM-1.0)/2.0*EM**2 QRW 1260
TR=TINF*(1.0+EN) QRW 1270
FLAG=1.0 QRW 1280
19 IF(IBFLAG)1,20,1 QRW 1290
20 ALPHA=ALLOC-ALPH QRW 1300
12 ALPHA=-ALPH QRW 1310
14 IF(TABB(4))15,16,15 QRW 1330
15 BETA=ENTERP(ALPH,TABB) QRW 1340
IF(.NOT.SIGNAL) GO TO 7 QRW 1350
GO TO 1 QRW 1360
16 IF(ALPH)17,18,18 QRW 1370

```

ISN 0038	17 BETA=IAB(3)*ALPH+1.0	QRW 1380
ISN 0039	GO TO 1	QRW 1390
ISN 0040	18 BETA=IAB(5)*ALPH+1.0	QRW 1400
ISN 0041	1 IF((EM.EQ.0.).AND.(A.EQ.0.).AND.(B.EQ.0.)) GO TO 77	QRW 1410
ISN 0043	QRW=BETA*EK*(REXM*EM)**A*EM**B*(TR-T-459.69)/(TR-TINF)	QRW 1420
ISN 0044	GO TO 7	QRW 1430
ISN 0045	77 QRW=0.	QRW 1440
ISN 0046	7 RETURN	QRW 1450
ISN 0047	END	QRW 1460